Computer Science Principles

Web Programming

Web-Presentation Design with CSS

CHAPTER 11: CSS HIERARCHY AND SELECTORS

DR. ERIC CHOU IEEE SENIOR MEMBER

CSS3

Overview

LECTURE 1

CSS

What is CSS3?

- Cascading Style Sheets (CSS) is the W3C standard for defining the presentation of documents written in HTML, and in fact, any **XML** language .
- Presentation, again, refers to the way the document is displayed or delivered to the user, whether on a computer screen, a cell phone display, printed on paper, or read aloud by a screen reader.
- With style sheets handling the presentation, HTML can handle the business of defining document structure and meaning, as mentioned.
- Public resource that can use:
  - http://www.dynamicdrive.com/style/
  - http://www.free-css-templates.com/

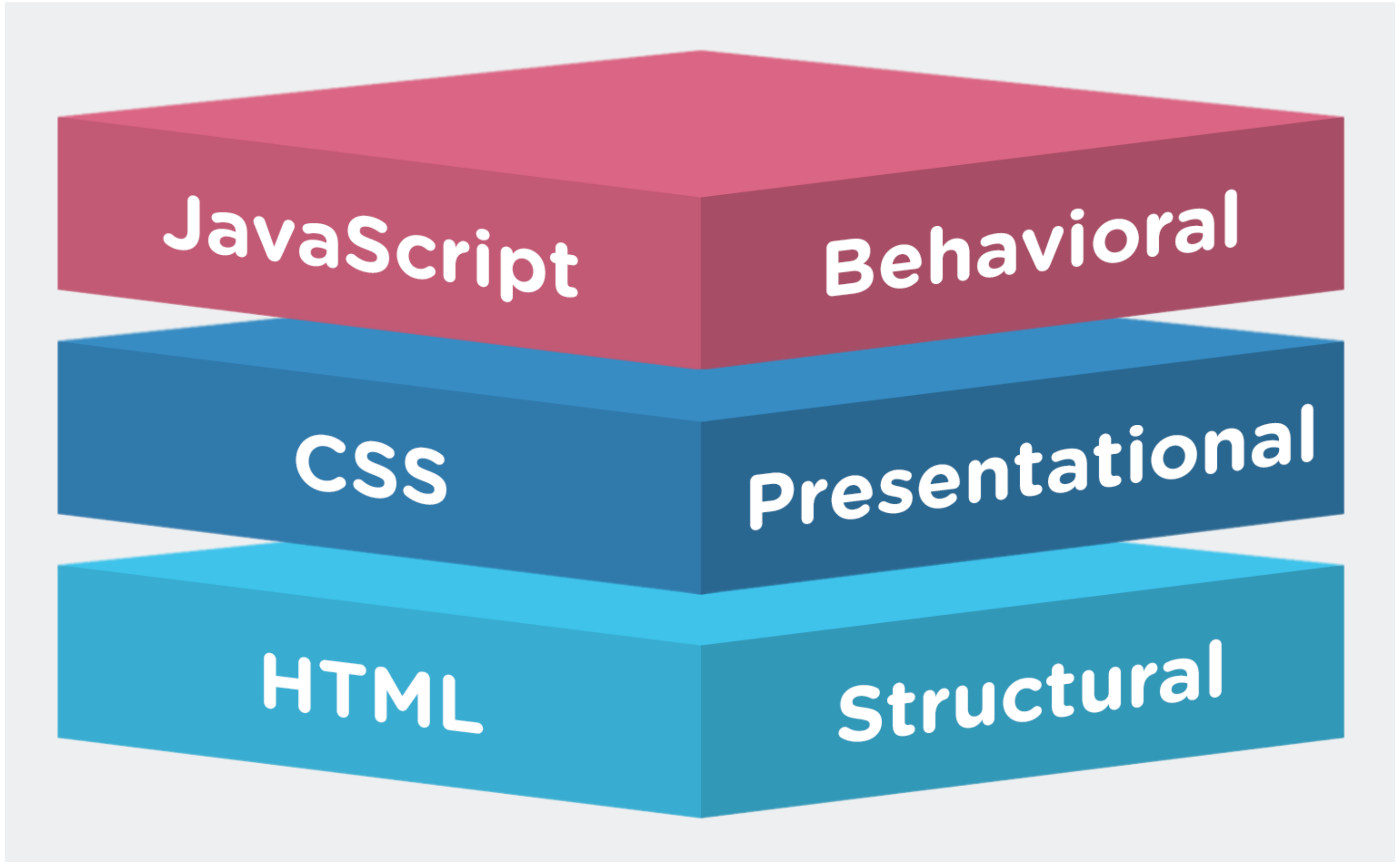
The Benefits of CSS

- Precise type and layout controls
- Less work
- More Accessible sites
- Reliable browser support

How Style Sheets Work

- Marking up the document
- Writing the rules
- Attaching styles to the document

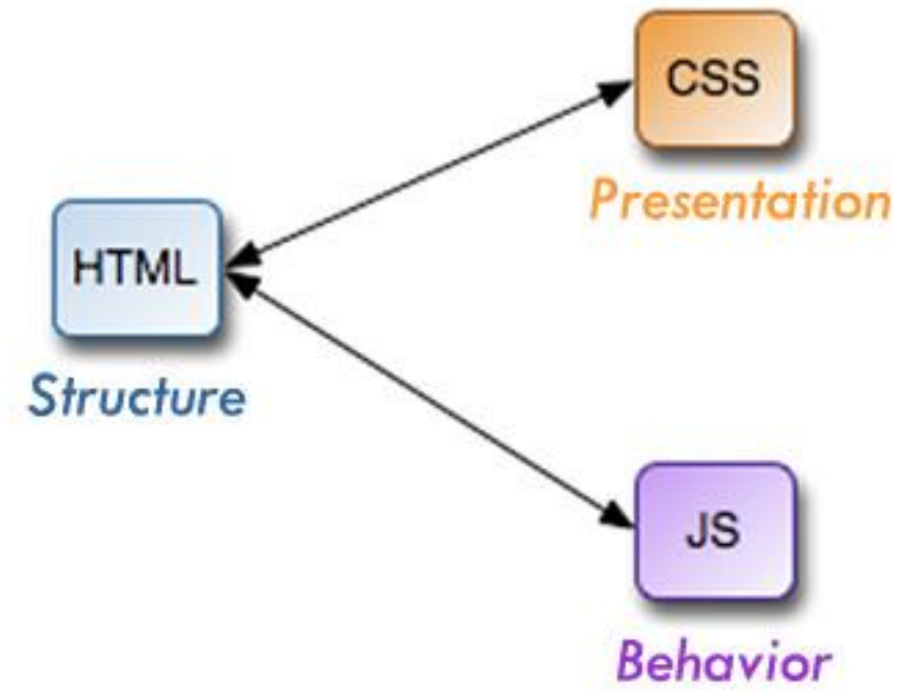
Marking up the document

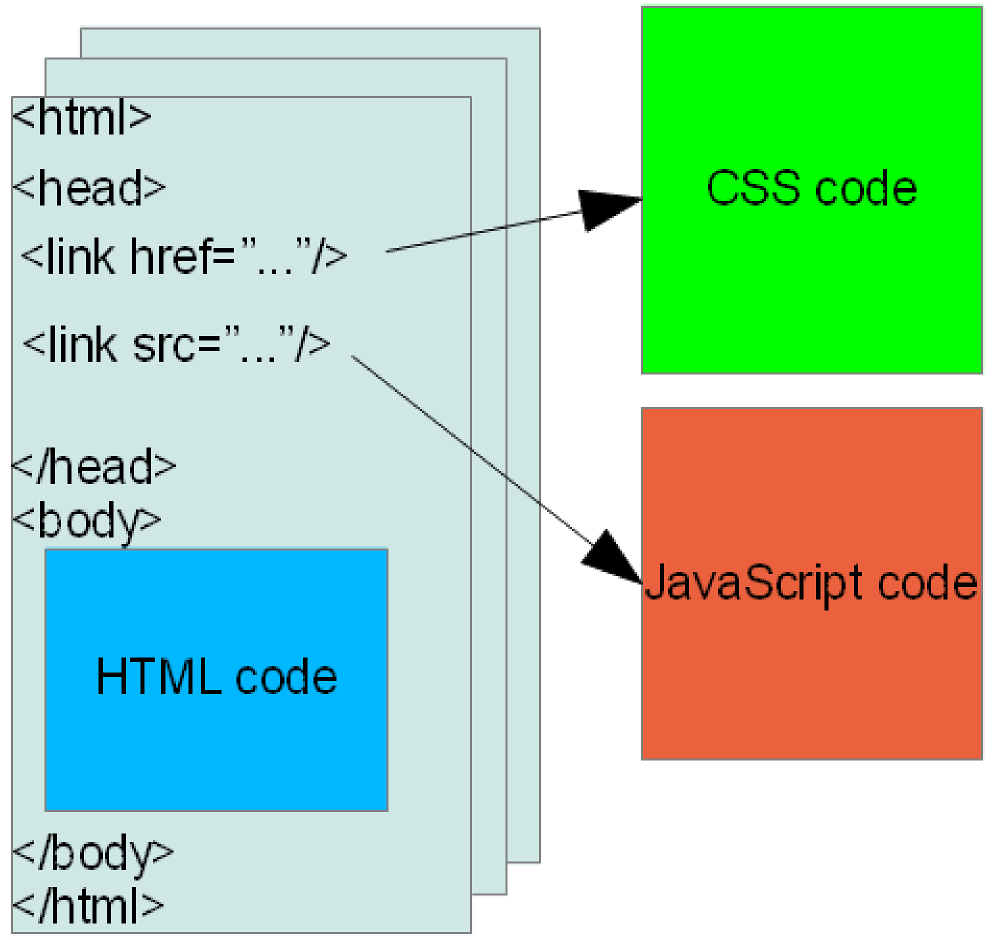


Object Hierarchy

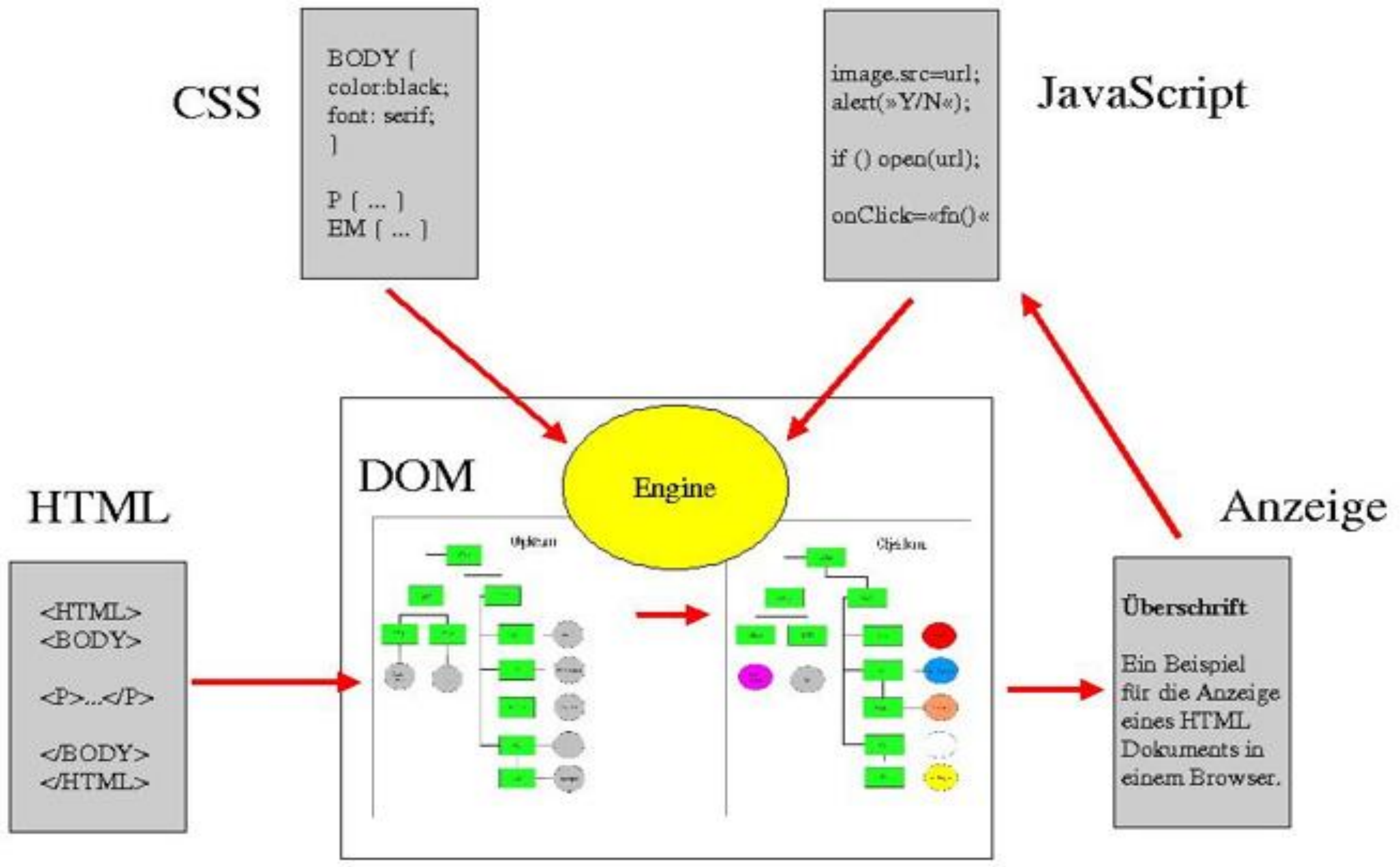
LECTURE 2

How DOM Work for HTML/CSS/JavaScript?





HTML, CSS, JavaScript und DOM



```
BODY {
color:black;
font: serif;
}

P { ... }
EM { ... }
```

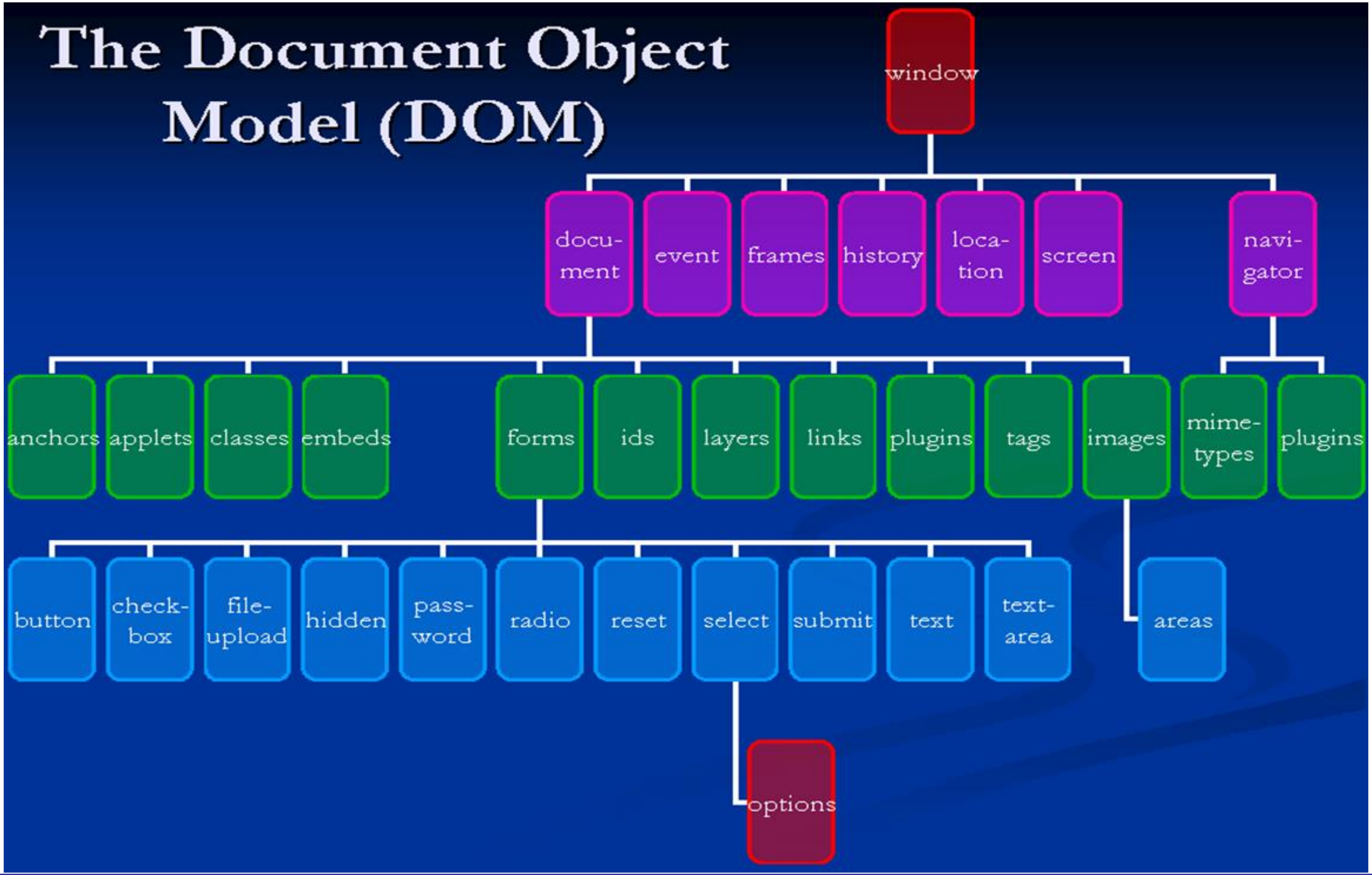
```
image.src=url;
alert(»Y/N«);

if () open(url);

onClick=«fn()«
```

```
<HTML>
<BODY>

<P>...</P>

</BODY>
</HTML>
```

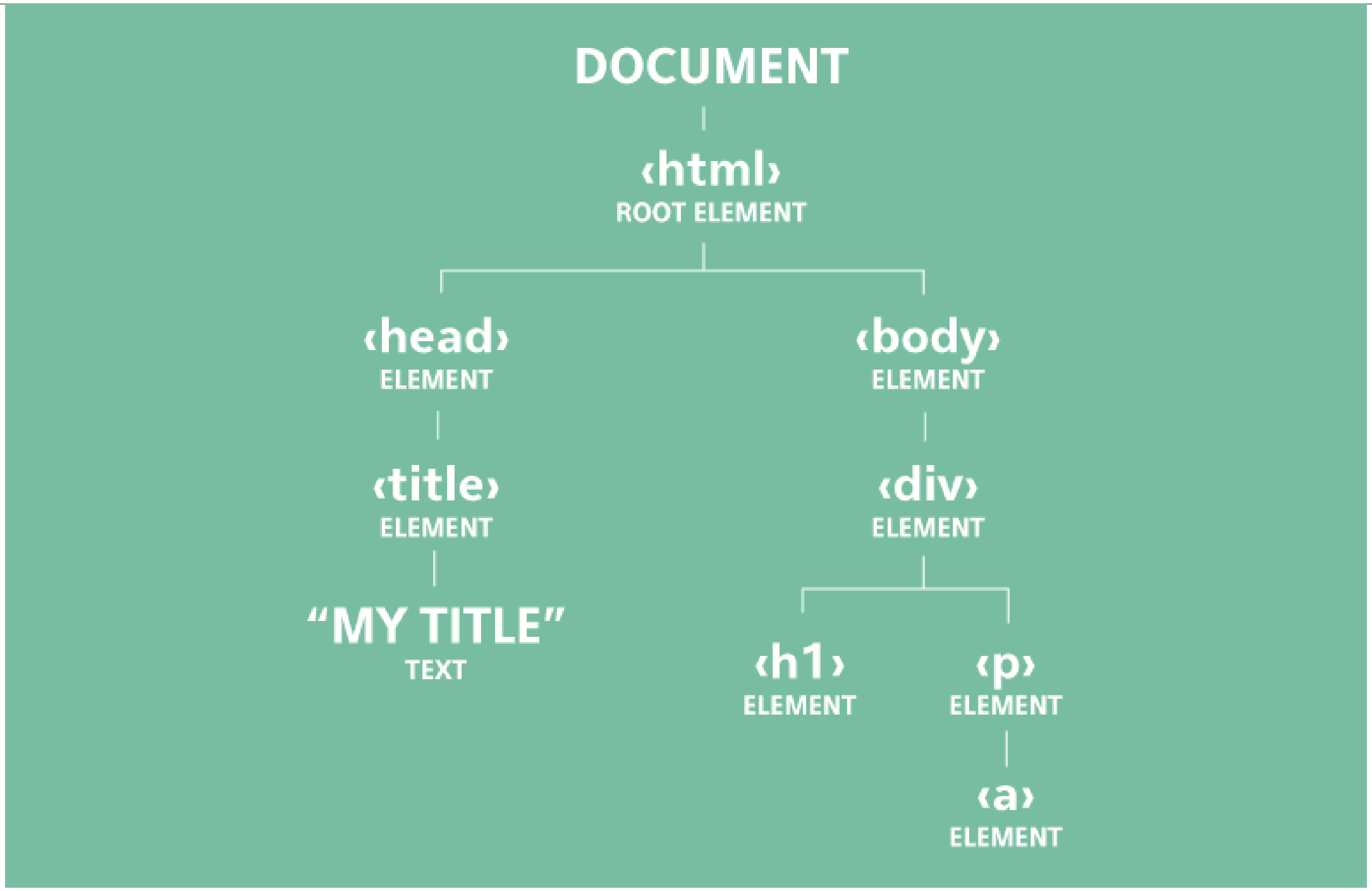
Überschrift

Ein Beispiel für die Anzeige eines HTML Dokuments in einem Browser.

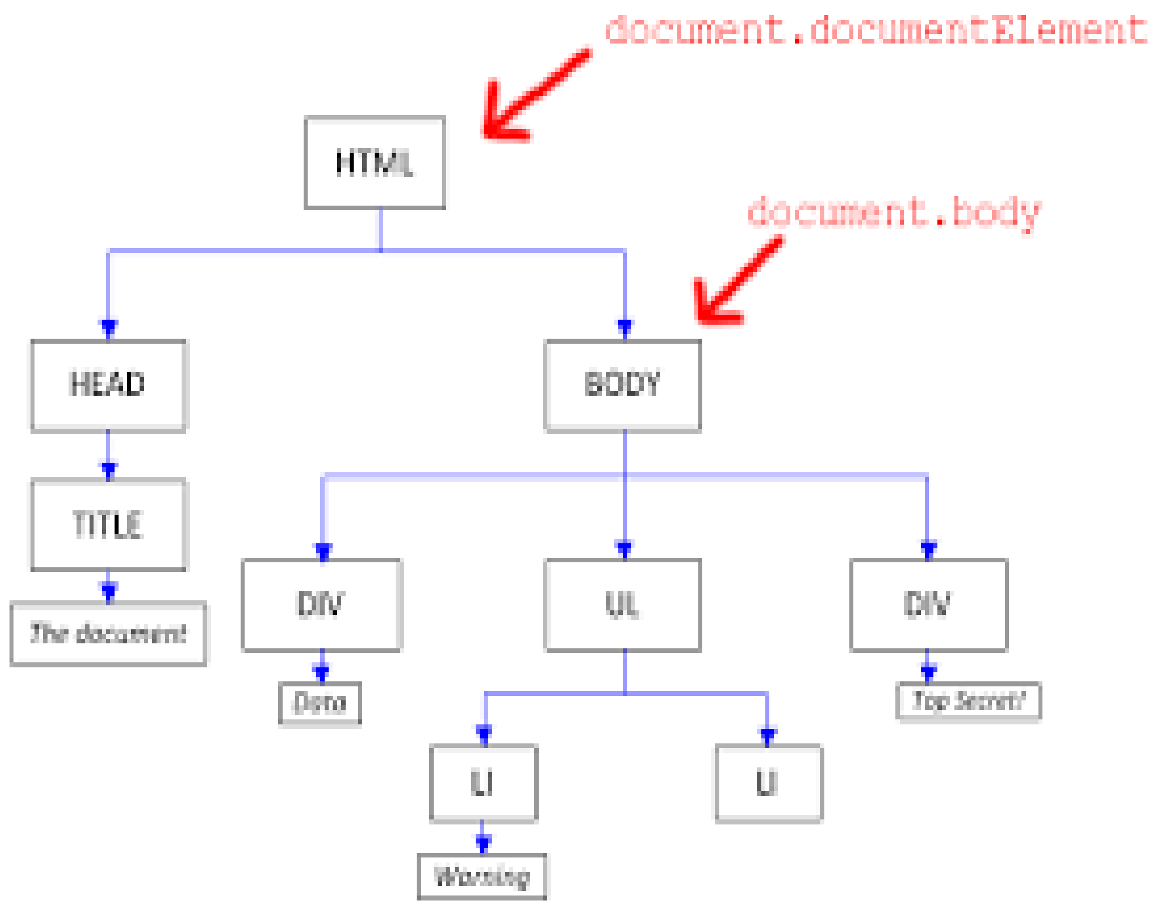
The Document Object Model (DOM)

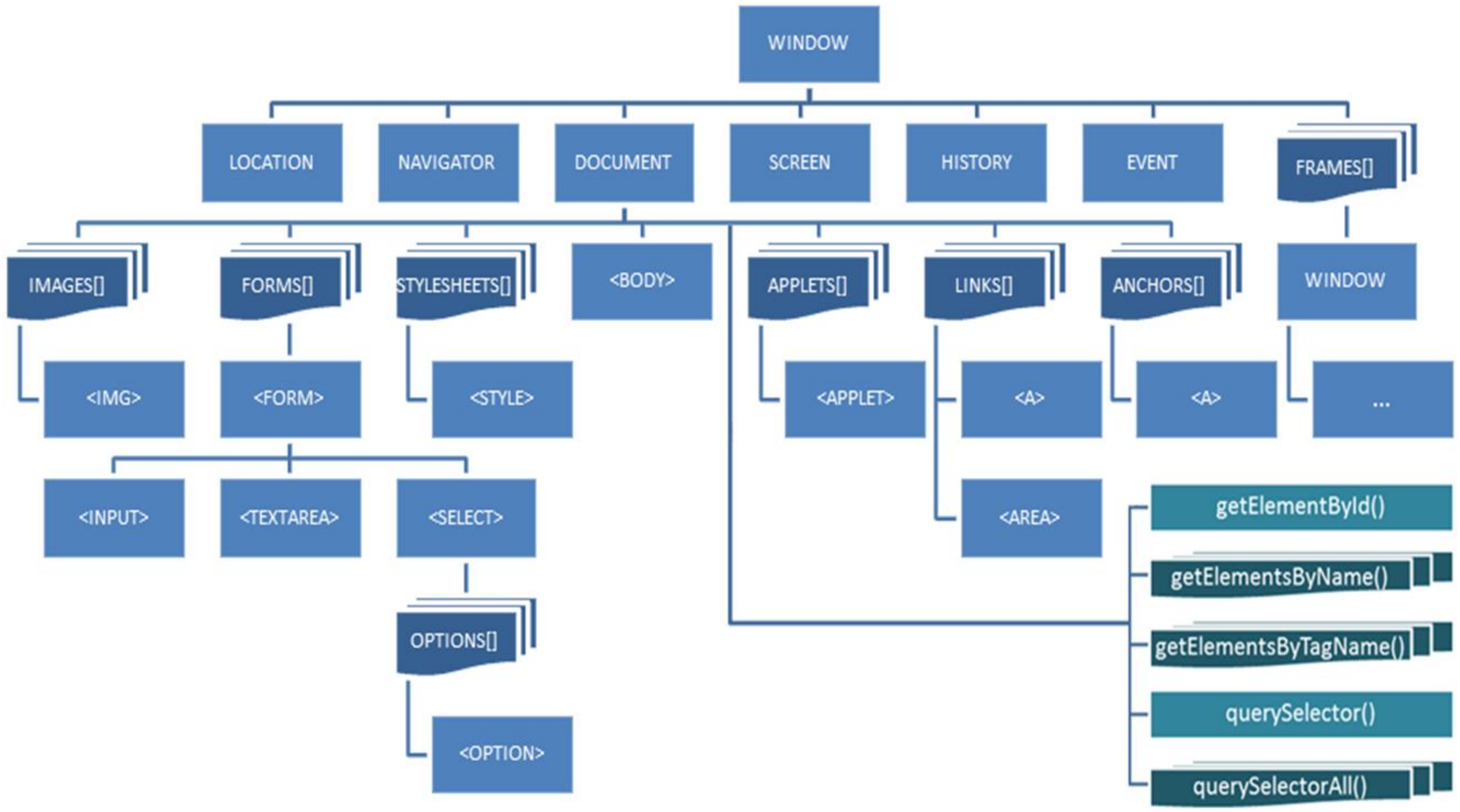


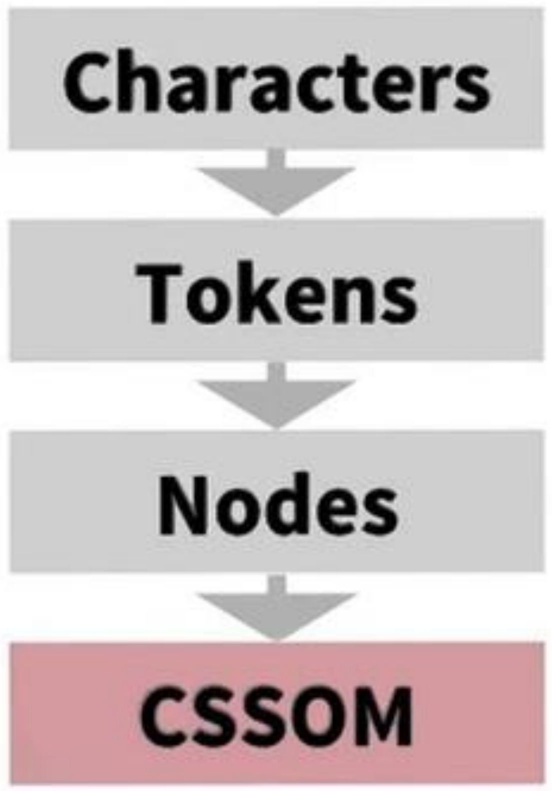
Structural View by HTML



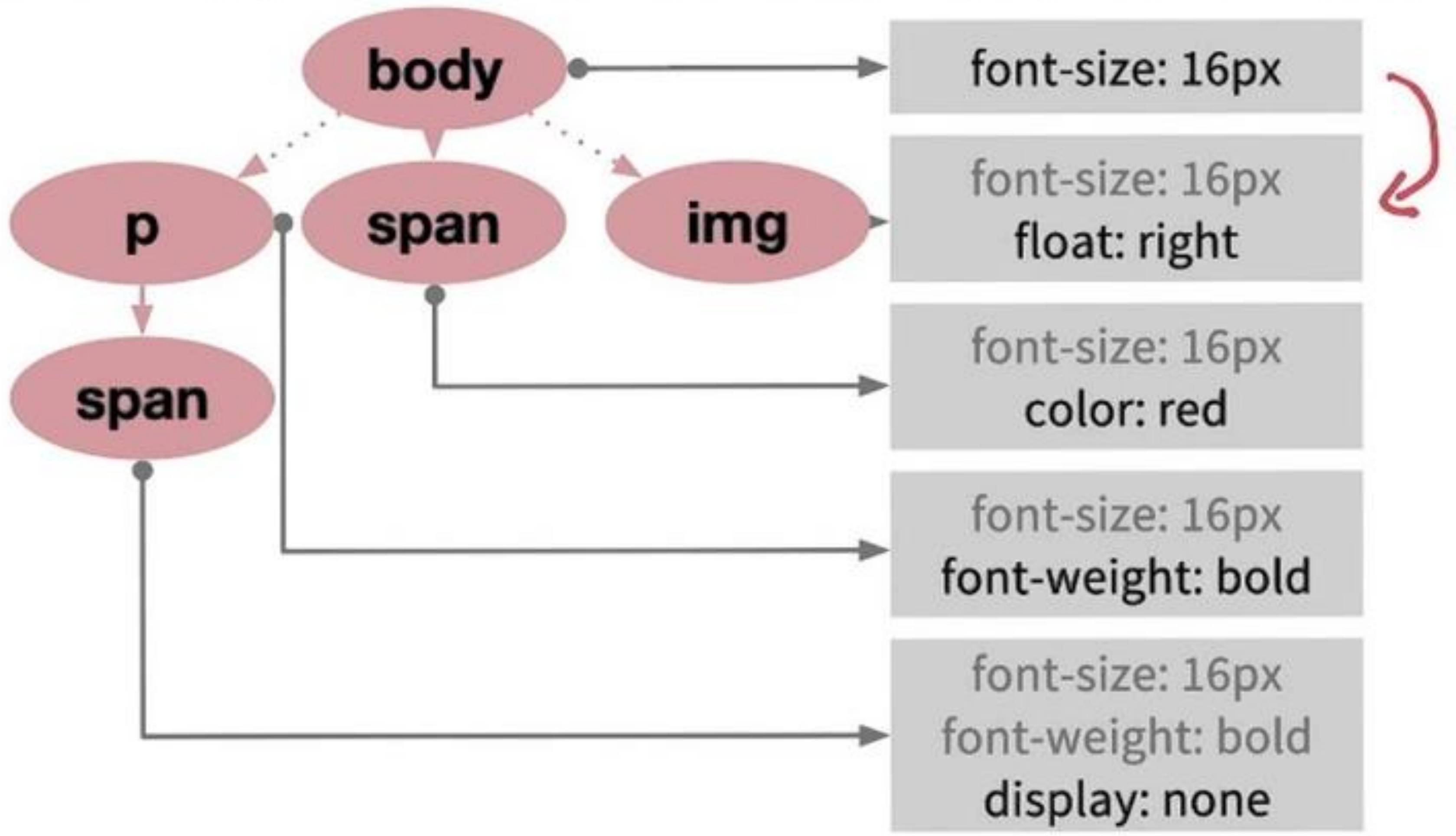
Structural View by JavaScript




WINDOW
LOCATION
NAVIGATOR
DOCUMENT
SCREEN
HISTORY
EVENT
FRAMES[]
IMAGES[]
FORMS[]
STYLESHEETS[]
<BODY>
APPLETS[]
LINKS[]
ANCHORS[]
WINDOW
<IMG>
<FORM>
<STYLE>
<APPLET>
<A>
<A>
...
<INPUT>
<TEXTAREA>
<SELECT>
<AREA>
OPTIONS[]
<OPTION>
getElementById()
getElementsByName()
getElementsByTagName()
querySelector()
querySelectorAll()

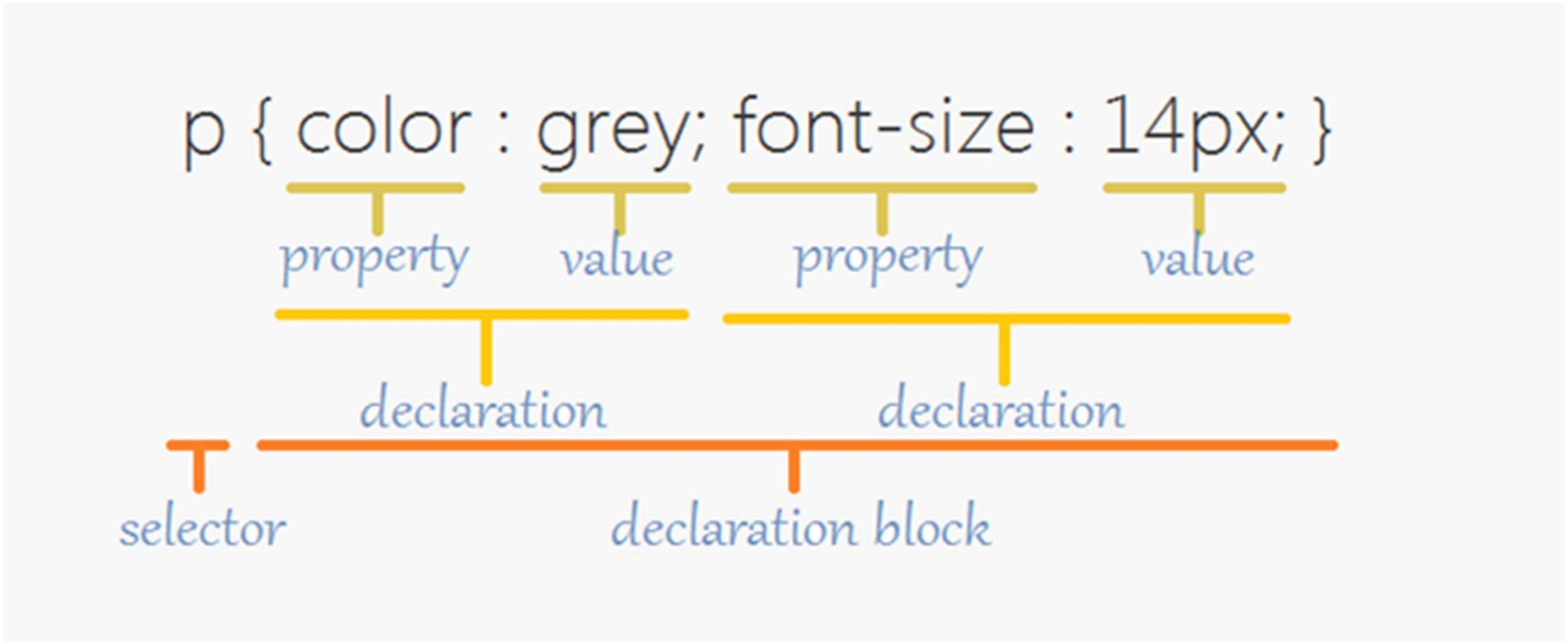


```
body {font-size: 16px} p {font-weight: bold} span
{color: red} p span {display: none} img {float: right}
```

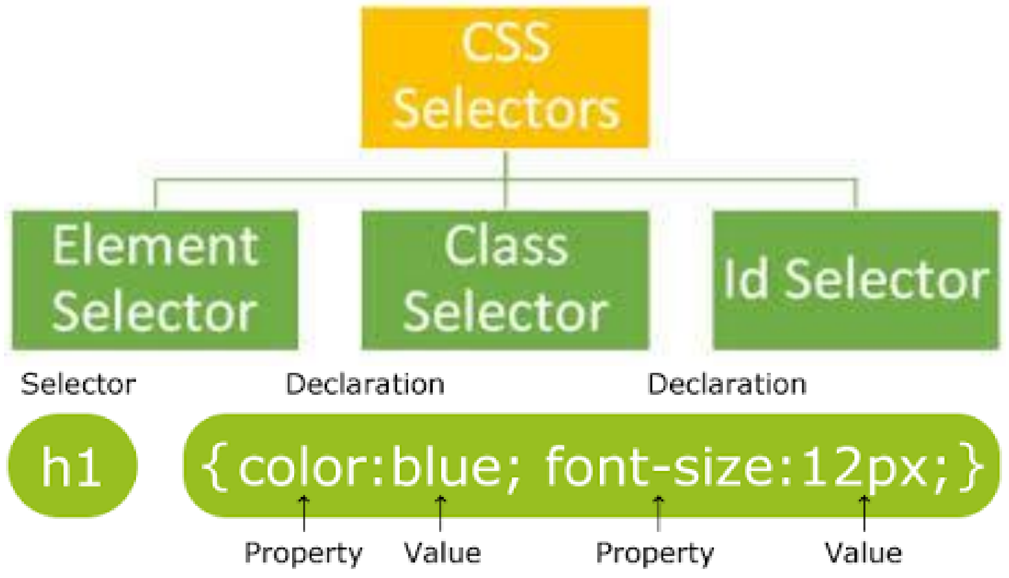


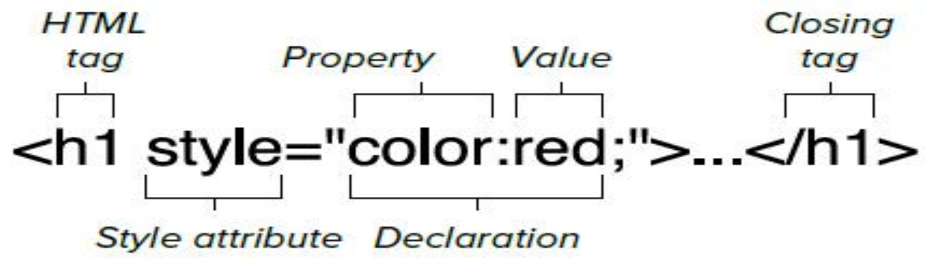
Writing CSS

LECTURE 3


p { color : grey; font-size : 14px; }
property
value
property
value
declaration
declaration
selector
declaration block

Structural View by CSS





The general syntax for defining styles directly in an HTML tag.

- Element selector:

The element selector selects elements based on the element name.

- Example:

```
<p > This is a paragraph </p>

p {
      text-align:center;
      color:red;
}
```

You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

- Class selector:

The class selector finds elements with the specific class.

- Example:

```
.center {
   text-align: center;
   color: red;
}
```

You can specify that only specific HTML elements should be affected by a class.

- Id selector:

The id selector uses the id attribute of an HTML tag to find the specific element.

- Example:

```
#para1 {
   text-align: center;
   color: red;
}
```

The style rule below will be applied to the HTML element with id="para1":

Writing CSS Style

- **Selectors**: the selector shown above is called Element Selector which means the style definition is for all <p> tags.
- **Declaration**: each property:value pair is called a declaration. A style definition could include several declarations which is named as declaration block.
- **Elements in a Element**:

  ```
  ul li { ... Style Definition ...}
  ```
- **Elements in a Element with id**:

  ```
  #unorderedlistitem li {… Style Definition ….}
  ```

Writing CSS Style

- **Class (group of Elements) in an Element:**
  form.age {… Style Definition …}
- **CSS Selector Structure:**
  grandparent parent me son grandchild {… Style Definition …}
- Work on Ex. 11-1

Attaching the Styles to the Documents

External Style Sheets:

```
<LINK REL=StyleSheet HREF="style.css" TYPE="text/css" MEDIA=screen>
```

Embedded Style Sheets:

```
<STYLE TYPE="text/css" MEDIA=screen>
<!-- BODY  { background: url(foo.gif) red; color: black }
      P EM  { background: yellow; color: black }
      .note { margin-left: 5em; margin-right: 5em } -->
</STYLE>
```

Inline Style Definition:

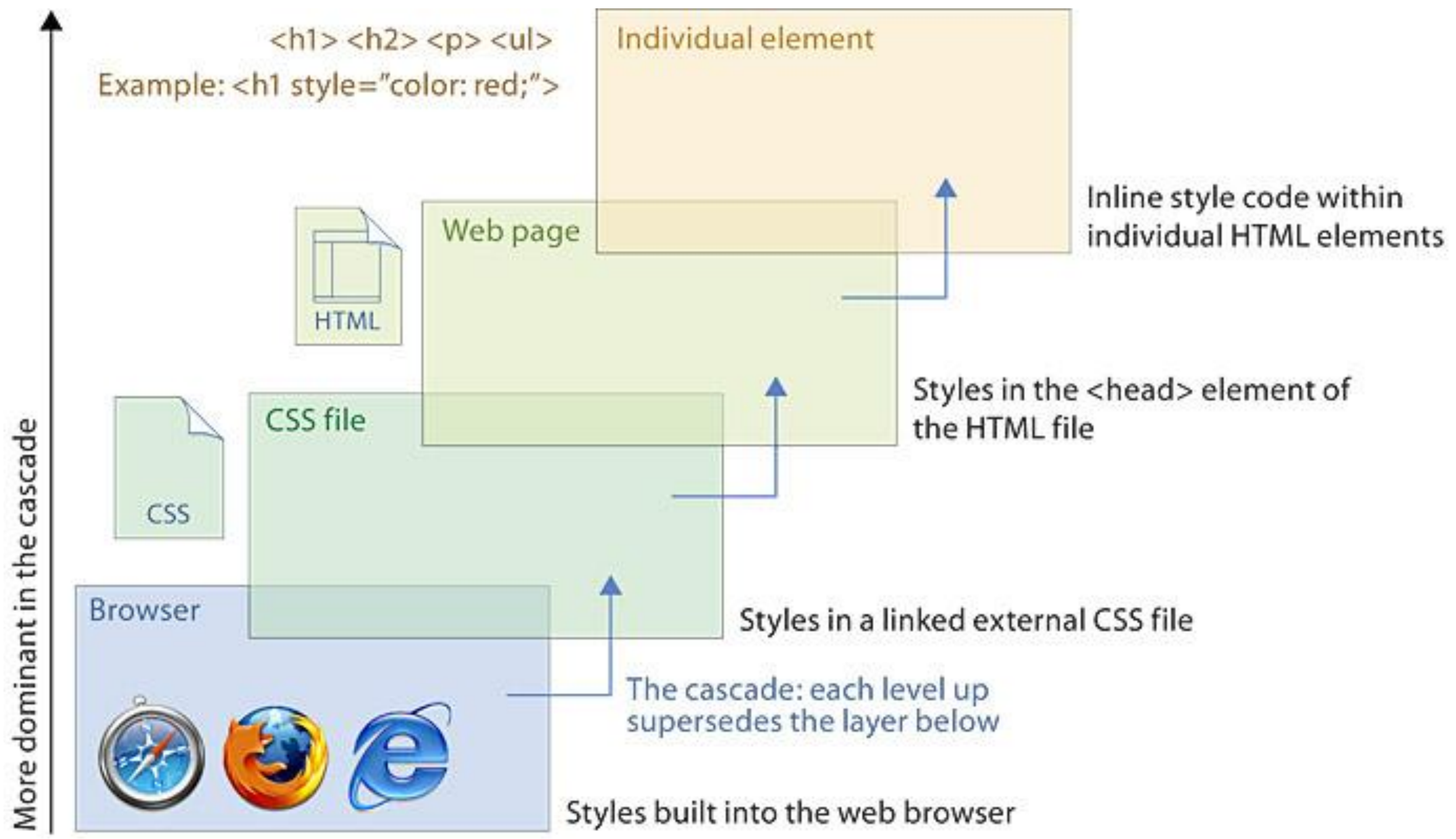
```
<P STYLE="color: red; font-family: 'New Century Schoolbook', serif"> This paragraph is
styled in red with the New Century Schoolbook font, if available.</P>
```

Attaching the Styles to the Documents

Imported Style Sheets:

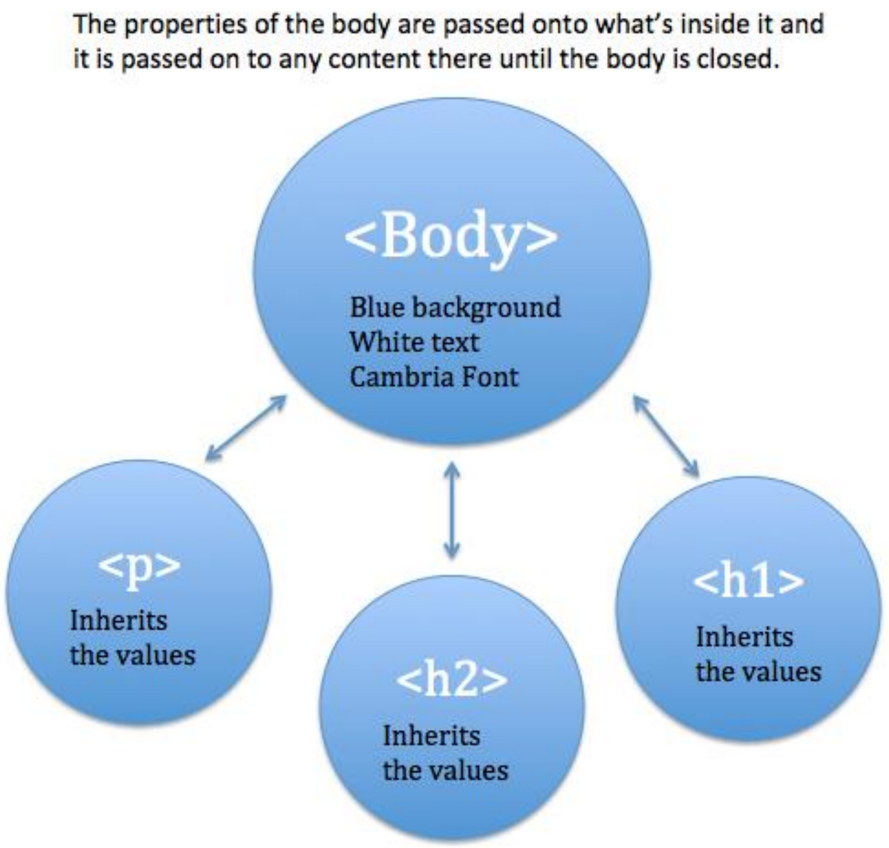
```
<STYLE TYPE="text/css" MEDIA="screen, projection">
<!--  @import url(http://www.htmlhelp.com/style.css);
      @import url(/stylesheets/punk.css);
      DT { background: yellow; color: black }-->
</STYLE>
```

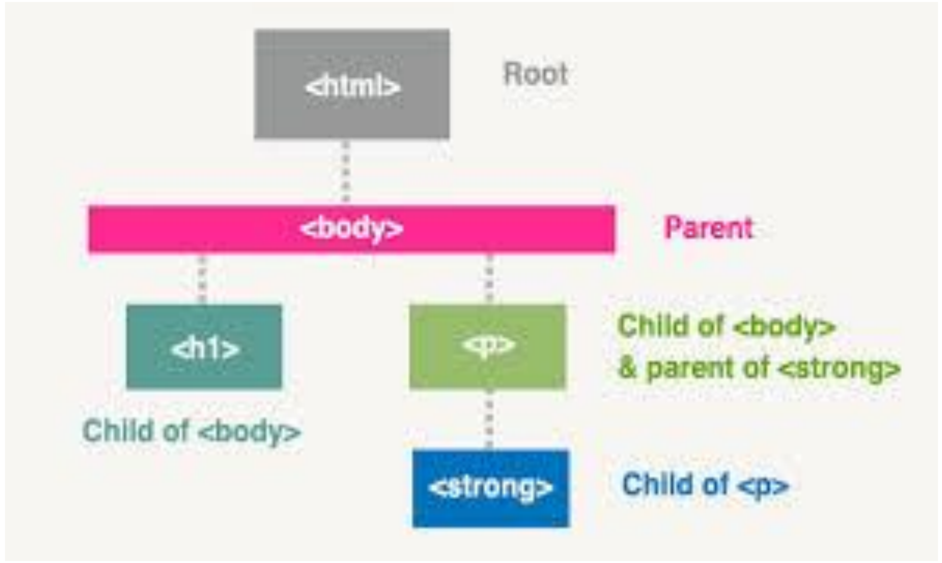
Where You may get CSS Definition



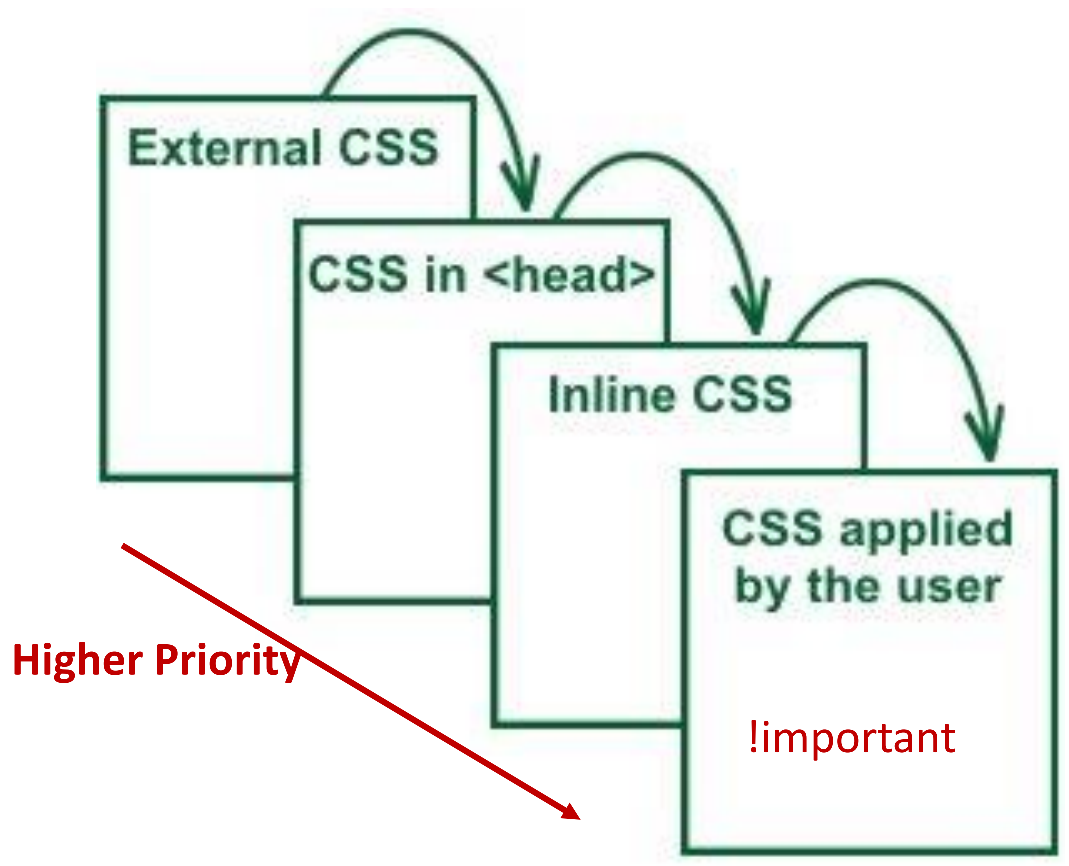
<h1> <h2> <p> <ul>
Example: <h1 style="color: red;">
Individual element
Inline style code within individual HTML elements
HTML
Web page
Styles in the <head> element of the HTML file
CSS
CSS file
Styles in a linked external CSS file
Browser
The cascade: each level up supersedes the layer below
Styles built into the web browser
More dominant in the cascade

Inheritance of Style Definition From Parents Unless Specified Otherwise


The properties of the body are passed onto what's inside it and it is passed on to any content there until the body is closed.
<Body>
Blue background
White text
Cambria Font
<p>
Inherits the values
<h2>
Inherits the values
<h1>
Inherits the values



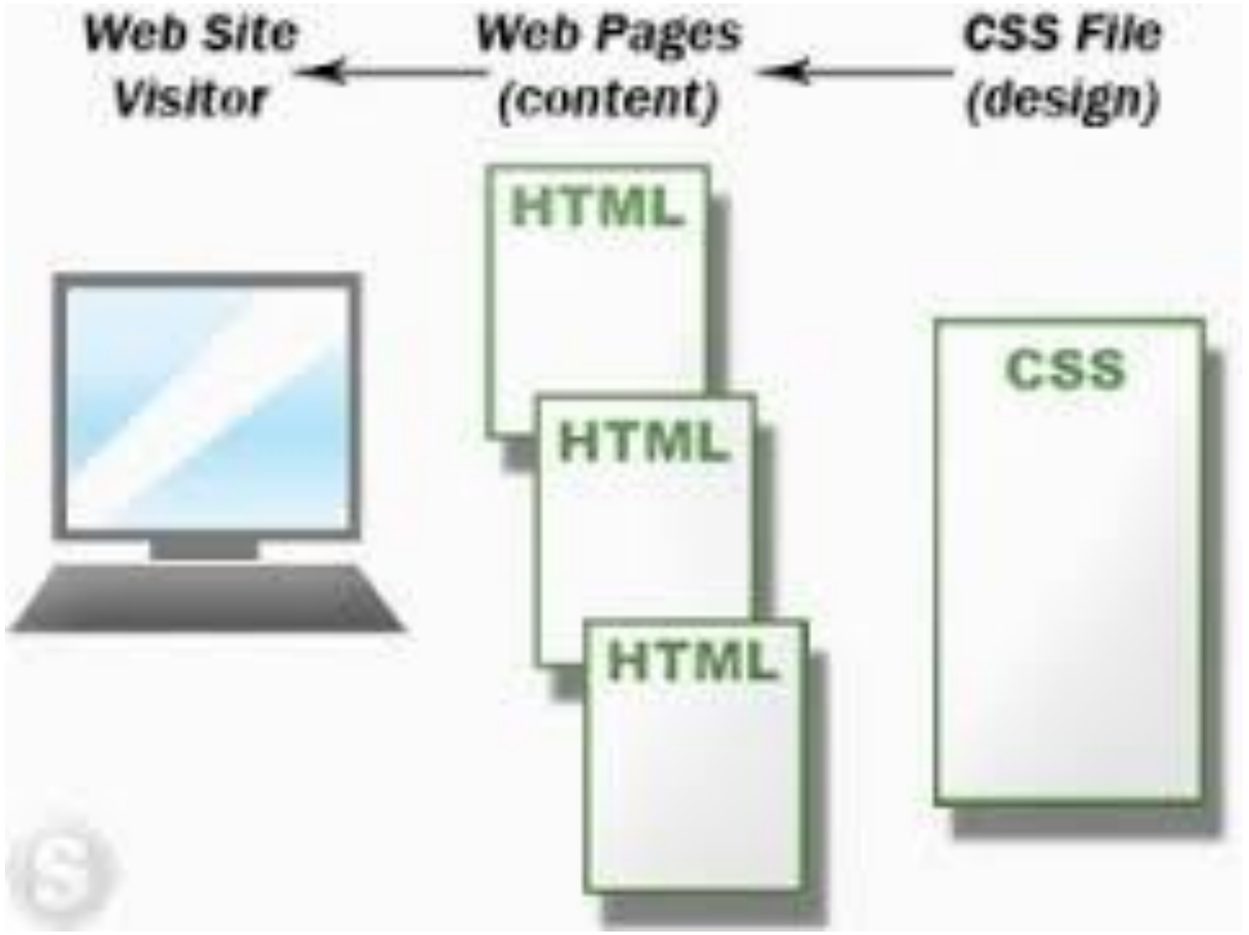
Cascading Means Put a Layer on top of Another



Style Sheet Hierarchy:

- Browser default settings (Weakest)
- User style settings (set in a browser as a reader style sheet: still kind of browser setting)
- Linked external Style Sheet <link rel=… href=…>
- Imported Style Sheets @import url(…)
- Embedded Style Sheets <style></style>
- Inline Style information <p style="…">
- Any Style Rule marked !important **by author**
- Any Style rule marked !important by **the reader (user)**

Cascading (2nd Meaning) One Sheet and for All



CSS

Inheritance and Cascading Priority

- The cascade refers to what happens when several sources of style information view for control of the elements on a page: style information is passed down **(“cascades” down)** until it is overridden by a style command with more weight. More weight means higher priority or lower in the element tree.
- !important is a indicator to prevent a specific rule from being overridden.

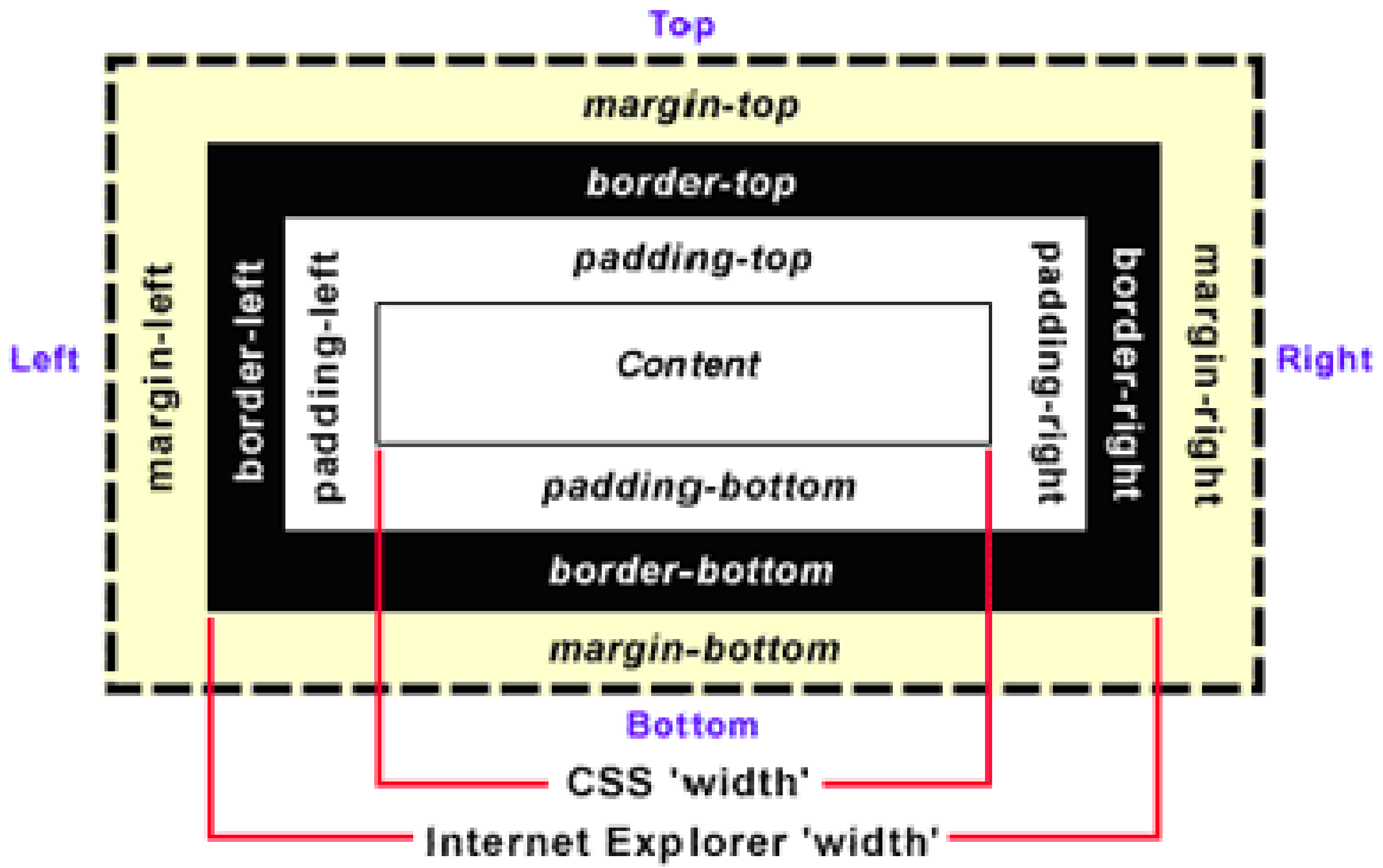
Grouped selectors (all selectors will be affected)

```
h1,h2,p,em,img { border: 1px solid blue;}
```

Box Model

LECTURE 4

CSS Box model
(What CSS will define for Styles)



Basic Selectors

Universal Selector:

* {   border: 1px solid #02C2D8;}

Element Selector:

li {   border: 2px solid #04B1D9;}

Class Selector:

.must-have {   border: 3px solid #0587BF;}

ID Selector:

#wants {   border: 4px solid #0668A4;}

Grouping CSS Selector:

#content, #needs, p {   border: 2px solid #20CC80; /*green*/

.want-to-have, #wants {   border: 2px solid #3D4173; /*purple*/}

Advanced Selectors

Example used for Advanced Selectors

```
<div id="content">
    <ul id="needs" class="needs">
            <li id="one" class="must-have">House</li>
            <li id="two" class="must-have">Land</li>
            <li id="three" class="must-have">Car</li>
     </ul>
     <div>
            <p class="qoute">Some essentials in life.</p>
            <div>
                <ul id="wants">
                        <li class="want-to-have">Mansion</li>
                        <li class="want-to-have">Ferrari</li>
                        <li class="want-to-have">Kingdom</li>
               </ul>
               <p class="qoute">Teach a man to fish and that makes you an awesome person.</p>
            </div>
     </div>
</div>
```

Graphical Representation of Element Descendants

```
<div id="qoute">
      <ul id="needs" class="needs">
            <li class="must-have">House</li>
            <li class="must-have">Land</li>
            <li class="must-have">Car</li>
      </ul>
</div>
```

I am a descentant of

<div id="qoute">

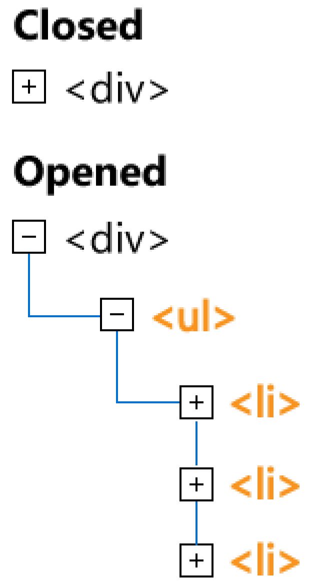
We are descentants of

<ul id="needs" class="needs">

<div id="qoute">

Indentation is a simple way to identify descendants of an element. The **<ul>** and **<li>** elements are descendants of the **<div>**.

Hint : Indent your code.



This representation shows all the elements that are contained inside another element are the descendants. **Descendants coloured orange.**

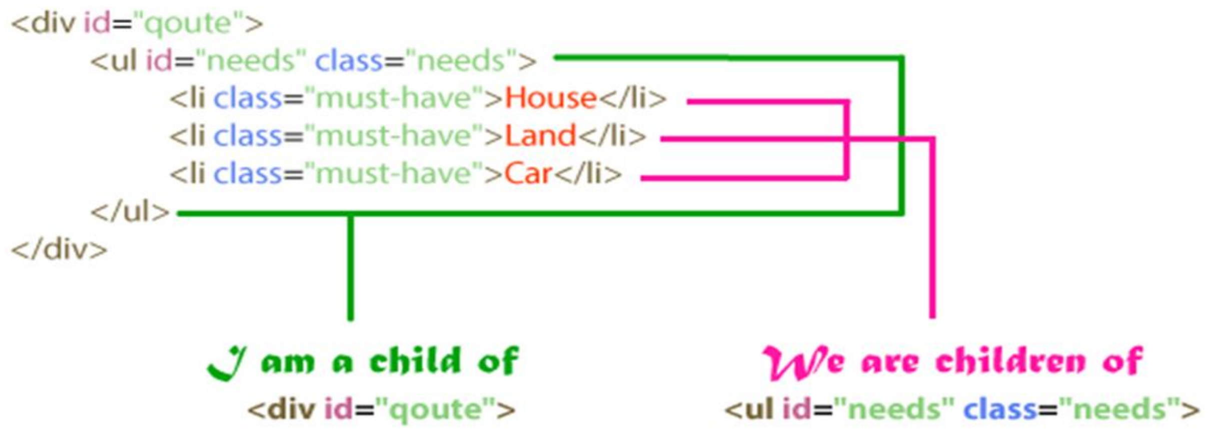
To Set Style for all descendents of ul id="needs" and including

- The syntax for a descendant combinator selector is the parent or ancestor followed by a whitespace then the descendant.

```
#needs li {
    border: 1px solid #02C2D8;
}
```

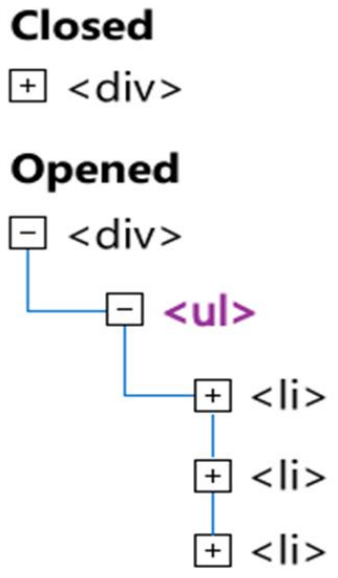
- Only the <li> elements under #needs

Graphical Representation of Child Elements



```
<div id="qoute">
    <ul id="needs" class="needs">
        <li class="must-have">House</li>
        <li class="must-have">Land</li>
        <li class="must-have">Car</li>
    </ul>
</div>
```

The child has a direct relationship with its parent. The **<div>** has a **<ul>** as a child and the **<ul>** has **<li>** elements as children but the **<li>** elements are not direct children of **<div>**.



The Lone Child

This story follows the journey of a **<div>** who has only one **purple** child named **<ul>**. Though **<ul>** has children of his own, they are not children of **<div>**.

To set the Style for all of the Children for <div id="quote">

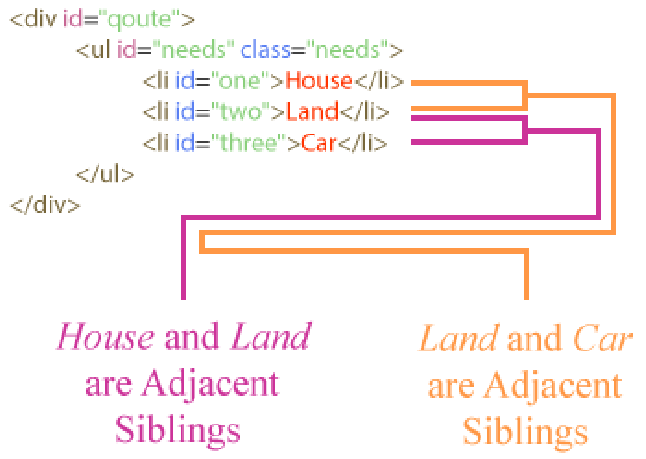
The syntax for a child combinator selector is the parent followed by a greater than (>) sign then the child.

```
#quote > ul {
    border: 2px solid #04B1D9;
}
```

In this example, it only has a child <ul>, but it can be of any type of elements such as <textarea> and etc.

Graphical Representation of Sibling Elements

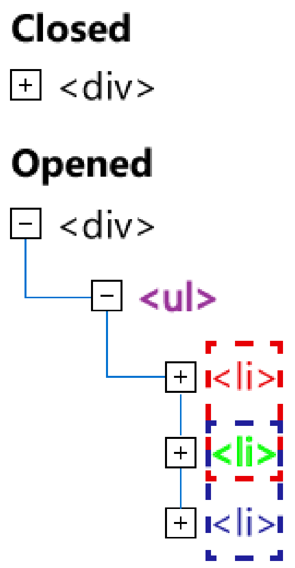
Adjacent Siblings



```
<div id="qoute">
        <ul id="needs" class="needs">
                <li id="one">House</li>
                <li id="two">Land</li>
                <li id="three">Car</li>
        </ul>
</div>
```

House and *Land* are Adjacent Siblings
Land and *Car* are Adjacent Siblings

Adjacent Siblings



Caring Siblings

Three **<li>** siblings. Red, Green and Blue. They all together love each other, especially Green it's true.

Red is adjacent to Green and Green is adjacent to Blue.

Forever and ever they stay together even though Red's not adjacent to Blue.

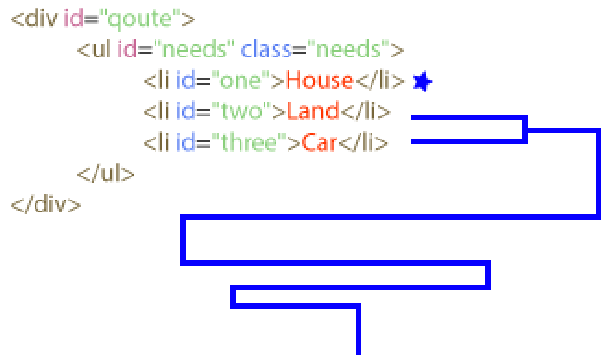
Set the Style for the next Sibling for <li id="one">

- The syntax for the adjacent sibling combinator is the first element in the selector sequence followed by the plus (+) sign then the second element that immediately succeeds the first.

```
#one + li {
    border: 3px solid #06BD06;
}
```

Graphical Representation of Sibling Elements

General Siblings



```
<div id="qoute">
        <ul id="needs" class="needs">
                <li id="one">House</li> ★
                <li id="two">Land</li>
                <li id="three">Car</li>
        </ul>
</div>
```

Land and *Car* are the general siblings of *House*.

The star (★) represents the element whose relationship is being described.
i.e The <li> containing text "House".

Set the Style of all of the Siblings of <li id="one">

- The syntax for the general sibling combinator is the first element in the selector sequence followed by the tilde (~) sign then a simple selector. i.e type, class, id etc.

```
#one ~ li {
    border: 3px solid #0F860F;
}
```

Making a Long Selector

- Here we will make a long and valid selector to see how much you have grasped. So far when using a class selector, every element with that class is matched. This is because we have not been specifying where in the DOM tree to look, so all is matched.

```
#quote > ul  ul > #one ~ li {
    border: 3px solid #FF6EC7;
}
```

Conclusion

- Combinators gives us the ability to go crazy with our selectors. Selectors can become long and complicated; it is good to practice using combinators to make lengthy selectors. This will show that you have a thorough understanding of what has been covered.