RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Internet and Web Technology**

Subject Code: **CS-504**

Semester: **5th**

Subject Notes

Subject code-CS 504 **Subject Name- Internet & Web Technology**

A Style Sheet is a collection of style rules that that tells a browser how the various styles are to be applied to the HTML tags to present the document. Rules can be applied to all the basic HTML elements, for example the tag, or you can define you own variation and apply them where you wish to.

There are three types of Style Sheets:

- Embedded: the style rules are included within the HTML at the top of the Web page - in the head.
- Inline: the style rules appear throughout the HTML of the Web page - i.e. in the body.
- Linked: The style rules are stored in a separate file external to all the Web pages.

3.1 NEED FOR CSS

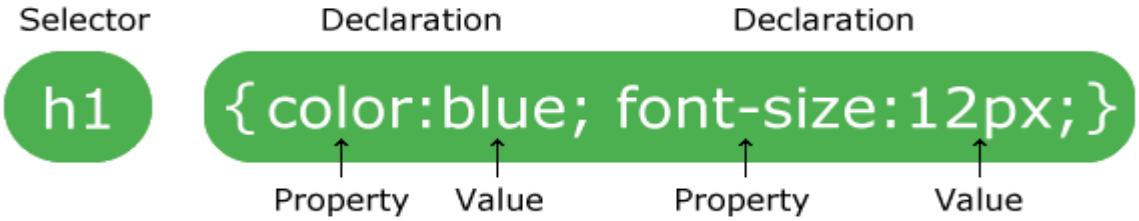
- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External style sheets are stored in CSS files

INTRODUCTION TO CSS

- HTML was NEVER intended to contain tags for formatting a web page!
- HTML was created to **describe the content** of a web page, like:
- <h1>This is a heading</h1>
- This is a paragraph.
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

3.2 BASIC SYNTAX AND STRUCTURE

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon. A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Example:

In this example all <p> elements will be center-aligned, with a red text color:

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

3.3 USING CSS BACKGROUND IMAGES

You can set background images in CSS using the background-image and several other properties to control the behaviour of the image. Background properties include: background-repeat. Background-attachment.

Possible values are:

- Top.
- Right.
- Bottom.
- Left. Enter.
- any combination of the above

CSS background-image Property

Example: Set a background-image for the <body> element:

```
body {
 background-image: url("paper.gif");
 background-color: #cccccc;
}
```

3.4 COLORS AND PROPERTIES

Adding color is done using the "color" property for the foreground color and the "background-color" property for the background color. The "color" property specifies the color of the text for the HTML element. There are 2 ways to specifycolor in CSS:

Example : Set the text-color for different elements:

```
body {
  color: red;
}

h1 {
  color: #00ff00;
}

p.ex {
  color: rgb(0,0,255);
}
```

3.5 MANIPULATATING TEXTS

This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified.

Text Color

The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Look at CSS Color Values for a complete list of possible color values.

Example :

```
body {
  color: blue;
}

h1 {
  color: green;
}
```

3.6 USING FONTS

CSS Fonts is a module of CSS that defines font-related properties and how font resources are loaded. It lets you define the style of a font, such as its family, size and weight, line height, and the glyph variants to use when multiple are available for a single character.

Difference between Serif and Sans-serif Fonts

Font Family

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

Example:

```
p {
  font-family: "Times New Roman", Times, serif;
}
```

3.7 BORDERS AND BOXES

By default in the CSS box model, the width and height you assign to an element is applied only to the element's content box. Border-box tells the browser to account for any border and padding in the values you specify for an element's width and height.

Example:

Include padding and border in the element's total width and height:

```
#example1 {
  box-sizing: border-box;
}
```

3.8 MARGINS

The margin property sets the margins for an element, and is a shorthand property for the following properties:

- margin-top
- margin-right
- margin-bottom
- margin-left

If the margin property has four values:

- margin: 10px 5px 15px 20px;
  - top margin is 10px
  - right margin is 5px
  - bottom margin is 15px
  - left margin is 20px

If the margin property has three values:

- margin: 10px 5px 15px;
  - top margin is 10px
  - right and left margins are 5px
  - bottom margin is 15px

If the margin property has two values:

- margin: 10px 5px;
  - top and bottom margins are 10px
  - right and left margins are 5px

If the margin property has one value:

- margin: 10px;
  - all four margins are 10px

Example

Set the margin for all four sides of a <p> element to 35 pixels:

```
p {
  margin: 35px;
}
```

3.9 PADDING LIST

The CSS padding properties are used to generate space around an element's content, inside of any defined borders. With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

CSS has properties for specifying the padding for each side of an element:

- padding-top

- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- Length - specifies padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Example: Set different padding for all four sides of a element:

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

3.10 POSITIONING USING CSS

The position property specifies the type of positioning method used for an element.
There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

Example

```
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
```

```
 height: 100px;
 border: 3px solid #73AD21;
}
```

3.11 CSS2

Cascading Style Sheets Level 2 (CSS2) is the second version of cascading style sheets developed by W3C. It's a declarative language used to enhance the hyper extensive text mark-up language. CSS2 is a subset of Cascading Style Sheets Level 1 and has enhanced capabilities like: Currently, W3C does not provide any CSS2 recommendations. CSS2 have backward compatibility, so all valid CSS1 is also valid CSS2.

- Media types concept
- Aural style sheets
- Features for internationalization
- Extended font selection
- Automatic numbering and generated content
- Cursors
- Dynamic outlines
- Capability to control content overflow, clipping
- Absolute, fixed and relative positioning
- Extended selector mechanism

3.12 OVERVIEW AND FEATURE CSS 3

- **Backward compatibility**. User agents supporting CSS2 will be able to understand CSS1 style sheets, while CSS1 user agents are able to read CSS2 style sheets and discarding parts they don't understand. Also, user agents with no CSS support will be able to view style-enhances documents. Of course,
- **Complementary to structured documents**. Style sheets complement structured documents (e.g. HTML and XML), providing stylistic information for the marked-up text. It should be easy to change the style sheet with little or no impact on the markup.
- **Vendor, platform and device independence**. Style sheets enable documents to be remaining vendor, platform and device independent. Style sheets themselves are also vendor and platform independent, but CSS2 allows you to target a style sheet for a group of devices (e.g. printers).
- **Maintainability**. By pointing to style sheets from documents, Webmasters can simplify site maintenance and retain consistent look and feel throughout the site.
- **Simplicity**. CSS2 is more complex than CSS1, but it remains a simple style language which is human read- and writable..
- **Network performance**. CSS provides for compact encodings of how to present content. Compared to images or audio files, which are often used by authors to achieve certain rendering effects, using style sheets will decrease the size of the content.

- **Flexibility**. CSS can be applied to content in several ways. The key feature is the ability to cascade style information specified in: the default UA style sheet, user style sheets,
- **Richness**. Providing authors with a rich set of rendering effects increases the richness of the Web as a medium of expression. Designers have been longing for functionality commonly found e.g. in desktop publishing and slide-show applications.
- **Alternate language bindings**. The set of CSS properties described in this specification form a consistent formatting model for visual and aural presentations. This formatting model can be accessed through the CSS language, but bindings to other languages are also possible. For example, a JavaScript program may dynamically change the value a certain element's 'color''color' property.
- **Accessibility**. Last, but not least, using CSS will increase accessibility to Web documents. By retaining textual information in text form, both robots indexing Web pages and human users will have more options for digesting the content. Users can provide their personal style sheets if author-suggested style sheets hinder accessibility.

3.13 CLIENT SIDE SCRIPTING WITH JAVA SCRIPT

Server-side scripting is executed by a web server; client-side scripting is executed by a browser. Client-end scripts are embedded in a website's HTML mark-up code, which is housed on the server in a language that's compatible with, or compiled to communicate with, the browser.

JavaScript is a scripting language most often used for client-side web development. Client-side refers to operations that are performed by the client (in our case the client is the browser) in a client-server relationship. Despite the name, JavaScript is essentially unrelated to the Java programming language.

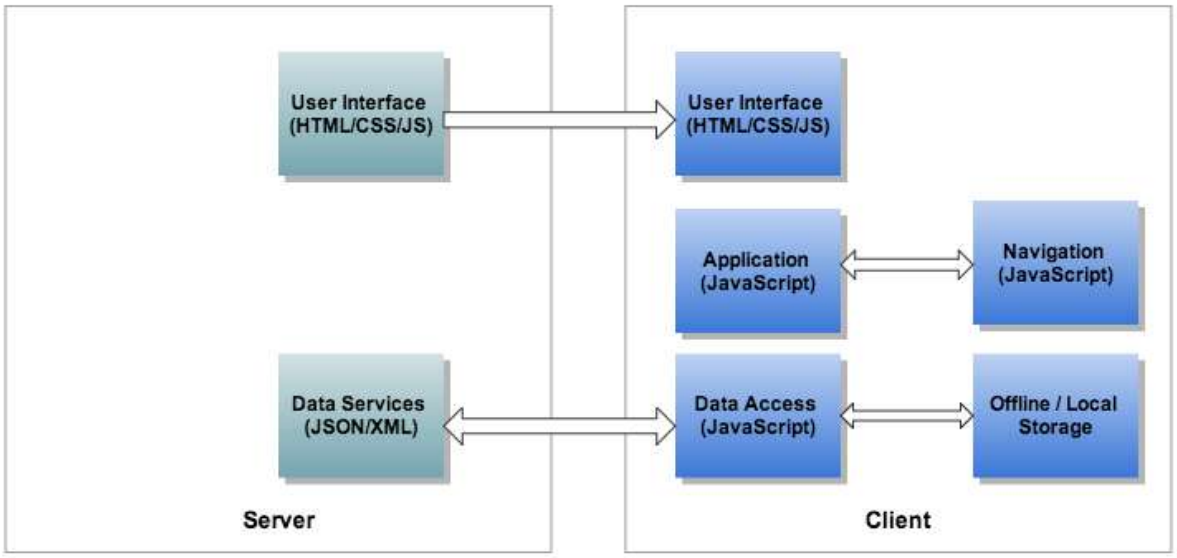


Figure 3.1 scripting language

Example:

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

3.14 VARIABLE, FUNCTION, CONDITION, LOOP AND REPETITIONS

3.14.1 VARIABLE

JavaScript variables are containers for storing data values.

In this example, x, y, and z, are variables:

Example

```
var x = 5;
var y = 6;
var z = x + y;
```

3.14.2 FUNCTION

Earlier in this tutorial, you learned that functions are **declared** with the following syntax:

```
function functionName(parameters) {
  // code to be executed
}
```

Example

```
function myFunction(a, b) {
  return a * b;
}
```

3.14.3 CONDITION

Very often when you write code, you want to perform different actions for different decisions.

In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

Example

```
if (hour < 18) {
  greeting = "Good day";
}
```

3.14.4 LOOP AND REPETITIONS

What we need is a generic solution for repeating code with control over how many times the code repeats. In JavaScript, this solution is provided in the form of something known as a loop. There are three kinds of loops we can use to repeat some code:

- for loops
- while loops
- do...while loops

Each of these three loop variations allow us to specify the code we want to repeat (aka loop) and a way to stop the repetition when a condition is met. In the following sections, we'll learn all about them.

Example:

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Loops!</title>
 <style>
 </style>
</head>
<body>
 <script>
   for (var i = 0; i < count; i++) {
     saySomething();
```

```
  }
  function saySomething() {
    document.writeln("hello!");
  }
 </script>
</body>
</html>
```

3.15 POP UP BOXES

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

window.alert("sometext");

The window.alert() method can be written without the window prefix.

Example

alert("I am an alert box!");

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

Syntax

window.confirm("sometext");

The window.confirm() method can be written without the window prefix.

Example:

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
```

```
 txt = "You pressed Cancel!";
}
```

3.16 ADVANCE JAVA SCRIPTS

JavaScript is one of the **3 languages** all web developers **must** learn:

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behaviour of web pages

3.17 JAVA SCRIPT AND OBJECTS

- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

Objects are Variables

JavaScript variables can contain single values:

Example

```
var person = "John Doe";
```

The values are written as **name : value** pairs (name and value separated by a colon)

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

3.18 JAVA SCRIPT OWN OBJECTS

In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods

3.19 THE DOM AND WEB BROWSER ENVIRONMENTS

The JavaScript specification calls that a host **environment**. A host **environment** provides platform-specific objects and functions additional to the language core. **Web browsers** give a means to control **web** pages. Node.js provides.

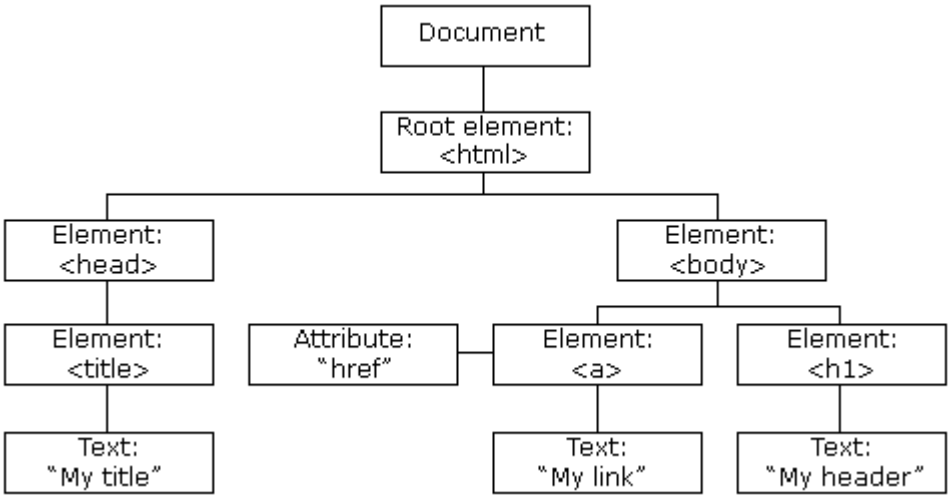


Figure 3.2 the dom and web browser environments

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

HTML DOM?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

3.20 MANIPULATION USING DOM

DOM manipulation methods allows you to add, edit or delete DOM element(s) in the web page. Use the selector to get the reference of an element(s) and then call manipulate methods to edit it. Important DOM manipulation methods: append(), propend(), before(), after(), remove(), replace All(), wrap() .

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements.

3.21 FORM AND VALIDATIONS

The data entered into a form needs to be in the right format and certain fields need to be filled in order to effectively use the submitted form. Username, password, contact information is some details that are mandatory in forms and thus need to be provided by the user.

Below is a code in HTML, CSS and JavaScript to validate a form:
1. HTML is used to create the form.
2. JavaScript to validate the form.
3. CSS to design the layout of the form.
Form validation :

```
<script>
function GEEKFORGEEKS()
{
   var name = document.forms["RegForm"]["Name"];
   var email = document.forms["RegForm"]["EMail"];
   var phone = document.forms["RegForm"]["Telephone"];
   var what =  document.forms["RegForm"]["Subject"];
```

```
var password = document.forms["RegForm"]["Password"];
var address = document.forms["RegForm"]["Address"];

if (name.value == "")
{
   window.alert("Please enter your name.");
   name.focus();
   return false;
}

if (address.value == "")
{
   window.alert("Please enter your address.");
   name.focus();
   return false;
}

if (email.value == "")
{
   window.alert("Please enter a valid e-mail address.");
   email.focus();
   return false;
}

if (email.value.indexOf("@", 0) < 0)
{
   window.alert("Please enter a valid e-mail address.");
   email.focus();
   return false;
}

if (email.value.indexOf(".", 0) < 0)
{
   window.alert("Please enter a valid e-mail address.");
   email.focus();
   return false;
}

if (phone.value == "")
{
   window.alert("Please enter your telephone number.");
   phone.focus();
   return false;
}

if (password.value == "")
{
```

```
    window.alert("Please enter your password");
    password.focus();
    return false;
  }

  if (what.selectedIndex < 1)
  {
    alert("Please enter your course.");
    what.focus();
    return false;
  }

  return true;
}</script>
```

3.22 DHTML

Dynamic HTML, or **DHTML**, is an umbrella term for a collection of technologies used together to create interactive and animated websites by using a combination of a static markup language (such as HTML), a client-side scripting language (such as JavaScript), a presentation definition language (such as CSS).

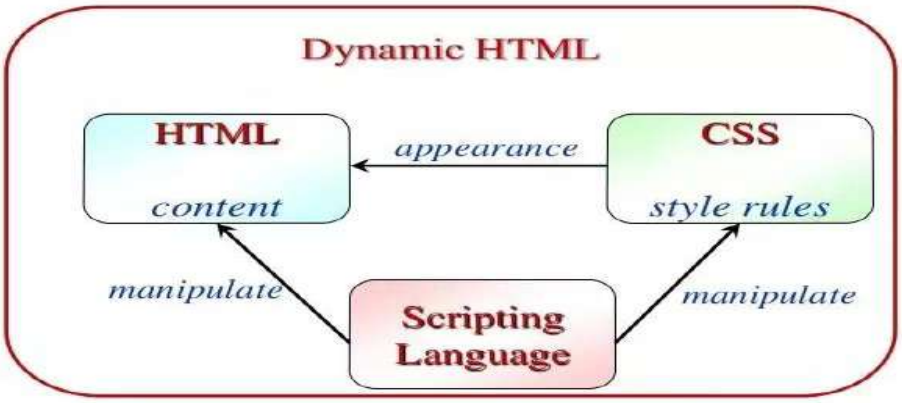


Figure 3.3 DHTML

3.23 Combining HTML

You can merge two or more table cells together by using the colspan attribute in a <td> HTML tag (table data). For example, in the below code is a table with three rows and three columns. If we wanted to combine the first two cells into one cell, we could use the colspan="2" attribute in the first <td> tag.

```
<table>
 <tr>
  <td colspan="2"> </td>
  <td> </td>
```

```
</tr>
<tr>
 <td> </td>
 <td> </td>
 <td> </td>
</tr>
<tr>
 <td> </td>
 <td> </td>
 <td> </td>
</tr>
</table>
```

3.24 CSS AND JAVASCRIPT

The <script> tag is used to define a client-side script (JavaScript).

The <script> element either contains script statements, or it points to an external script file through the src attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

To select an HTML element, JavaScript most often uses the document, get Element By Id () method.

This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

Example

```
<Script>
document .get Element ById("demo").inner HTML = "Hello JavaScript!";
</script>
```

3.25 EVENTS AND BUTTONS

An event handler allows you to execute code when an event occurs.

```
<h1 onclick="this.innerHTML='Ooops!'">Click on this text</h1>
```

you can also add a script in the head section of the page and then call the function from the event handler:

```
<html>
<head>
<script type="text/javascript">
function changetext(id)
{
```

```
id.innerHTML="Ooops!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">Click on this text</h1>
</body>
</html>
```

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in