

CS-504 (A) Internet and Web Technology

Unit – III (CSS & JS)

UNIT 03

Style sheets: Need for CSS, introduction to CSS, basic syntax and structure, using CSS, background images, colors and properties, manipulating texts, using fonts, borders and boxes, margins, padding lists, positioning using CSS, CSS2, Overview and features of CSS3

JavaScript: Client side scripting with JavaScript, variables, functions, conditions, loops and repetition, Pop up boxes, Advance JavaScript: Javascript and objects, JavaScript own objects, the DOM and web browser environments, Manipulation using DOM, forms and validations, DHTML : Combining HTML, CSS and Javascript, Events and buttons.

What is Script?

The term "scripting language" is also used loosely to refer to dynamic high-level general-purpose languages, such as JavaScript, VBScript, Perl, and Python, with the term "script".

- Scripting languages are becoming more popular due to the emergence of web-based applications.
- Also the increases in computer performance over the past few years has promoted a increase in the power and sophistication of scripting languages.

Major advantages of scripting languages:

1. Easy to learn and use
2. Minimum programming knowledge or experience required
3. Allow complex tasks to be performed in relatively few steps
4. Allow simple creation and editing in a variety of text editors
5. Allow the addition of dynamic and interactive activities to web pages
6. Editing and running code is fast.

Major Disadvantage of scripting languages

1. Because of code executes on the users computer, in some cases it can be exploited for malicious purposes. (Security Issues)
2. Not always able to work across different browsers. (Inconsistent)

Types of Scripts:

Scripts are Classified into the following two types:

1. Client Side Scripts
2. Server Side Scripts

1. Client Side Scripts

The script which is **running within the browser** is called as client side scripting.

Example:

1. Live Script
2. JavaScript
3. jQuery (JS Lib.)
4. AngularJS (Framework)
5. EmberJS (Framework)

2. Server Side Scripts

The Script which is **running within the web server** is called as server side scripting.

Example:

1. ASP ==> IIS (Internet Information Services)
 2. JSP ==> Tomcat/Sun Java System Web Server
 3. PHP ==> Apache
 4. W3C ==> Jigsaw Server
- etc

ASP, JSP, PHP, CGI, Perl, Ruby on Rails are the server side scripting technologies.

Difference between Scripting Languages and Programming Languages

Scripting Languages:

1. Interpreted based (Read Line by Line)
2. Implicit Declaration of data types

Example:

```
var x=100;
```

```
var str='Raju';
```

```
var x=100.000;
```

3. Limited Support for Application Development
4. Limited Support for Graphics Design or Game(s) Development
5. Easily Integrated with other Technologies

Programming Languages:

1. Compiler based
2. Explicit Declaration of data types

Example:

```
int a=100;
```

```
float b=200.00;
```

```
String s="java";
```

3. Rich Support for Application Development
4. Rich Support for Graphics Design or Game(s) Development
5. Difficult to Integrate with Other Technologies

Introduction to JavaScript

What is **JavaScript**?

JavaScript is a scripting language, that is, a lightweight programming language that is interpreted by the browser engine when the web page is loaded.

OR

JavaScript is the Scripting language of HTML and the Web.

JavaScript was created by Brendan Eich at Netscape and was first introduced in December 1995 under the name of LiveScript. It was rather quickly renamed JavaScript, although JavaScript's official name is ECMAScript (European Computer Manufacturer's Association) International organization.

Features of JavaScript

1. Generating HTML code on-the-fly without accessing the Web server
2. JavaScript can live by itself, Javascript is executed on the client side
3. Make your web pages responsive.
4. Detecting the user's browser, OS, screen size, etc.
5. Create cookies
6. Validate web form data.
7. Automatically change a formatted date on a Web page
8. Giving the user more control over the browser.
9. Handling dates and time.

Syntax

```
<Script type="text/javascript" language="javascript">  
statements;  
statements;  
statements;  
</Script>
```

```
<script>  
statements  
statements  
statements  
</script>
```

My First JavaScript Program

```
<!doctype html>  
<head>  
<script type="text/javascript" language="javascript">  
document.write("Welcome to JavaScript");  
</script>  
</head>
```

Save with.html or.htm Extension..!!

To print a statement using JS

```
<!doctype html>  
<head>  
<script type="text/javascript" language="javascript">  
  document.write("Welcome to JavaScript");  
  document.write('Welcome to the Future');  
</script>  
</head>
```

To provide space between two statements

```
<!doctype html>  
<head>  
<script type="text/javascript" language="javascript">  
document.writeIn("Welcome to JavaScript");  
document.write('Welcome to CDGI');  
</script>  
</head>
```

To change the line

```
<!doctype html>
<head>
<pre>
<script type="text/javascript" language="javascript">
document.writeln("Welcome to JavaScript");
document.write('Welcome to CDGI');
</script>
<pre>
</head>
```

You can also use
 tag

```
<!doctype html>
<head>
<script type="text/javascript" language="javascript">
document.writeln("Welcome to JavaScript <br/>"); right way
document.write('<br/>'); right way
<br/> //it is wrong (we can not write html tag inside JS code like this
document.write('Welcome to CDGI');
</script>
</head>
```

HTML tag inside JS

```
<!doctype html>  
<head>  
<script type="text/javascript" language="javascript">  
document.writeln("<h1> Welcome to JavaScript <h1/>");  
document.write('Welcome to CDGI');  
</script>  
</head>
```


CSS styling inside JS : To change colour of h1

```
<!doctype html>
<head>
<script type="text/javascript" language="javascript">
document.writeln("<h1 style='color:blue'> Welcome to JavaScript <h1/>");
document.write('Welcome to CDGI');
</script>
</head>
```

CSS styling inside JS : To change the font family or font size

```
<!doctype html>
<head>
<script type="text/javascript" language="javascript">
document.writeln("<h1 style='color:blue;font-family:tahoma;font-
size:20px'> Welcome to JavaScript <h1/>");
document.write('Welcome to CDGI');
</script>
</head>
```

What is DHTML?

When in side my webpage

- JavaScript is existed.
- HTML is existed.
- CSS is existed.

This is known as DHTML (Dynamic HTML)

- DHTML is an umbrella term.
- Collection of technologies that we are using.

Right way to use (“ ”) (‘ ’) Quotation marks

- Single (‘ ’) quotation inside double quotation (“ ”) inside and vice versa is **ok**.
- But double quotation (“ ”) is everywhere is **not ok**.

JavaScript Statements, Semicolons and Comments

JavaScript Statements, Semicolons and Comments

```
<!doctype html>  
<body>  
<script type="text/javascript" language="javascript">  
window.document.write("Window Object");  
document.write("document Object");  
</script>  
</body>
```

```
<!doctype html>  
<body>  
<script type="text/javascript" language="javascript">  
window.document.write("JavaScript");  
document.write("Live Script");  
</script>  
</body>
```

JavaScript Statements, Semicolons and Comments

```
<!doctype html>
<body>
<script type="text/javascript" language="javascript">
//document.write("JavaScript"); Single line comment
/*document.write("Live Script");
document.write("ECMA Script"); Multiline comment */
</script>
</body>
```

JavaScript Code, Block, Popup Boxes

JavaScript Code

JavaScript code is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they are written. This example will write a heading and two paragraphs to a web page:

Example

```
<script type="text/javascript">  
document.write("<h1>This is a heading</h1>");  
document.write("<p>This is a paragraph.</p>");  
document.write("<p>This is another paragraph.</p>"),  
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket {, and end with a right curly bracket). The purpose of a block is to make the sequence of statements execute together.

Example

```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph. </p>");
}
</script>
```

JavaScript Popup Boxes

JavaScript has three kind of popup boxes:

1. Alert box
2. Confirm box
3. Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("Message");
```

Example:

```
<html>  
<head>  
<title>Alert box</title>  
<script type="text/javascript">  
alert("Click OK to Proceed");  
alert("CDGI");  
</script>  
</head>  
<body>
```

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

```
confirm("Message");
```

```
<!doctype html>
```

```
<body>
```

```
<script type="text/javascript" language="javascript">
```

```
var MyResult=confirm("Click OK or CANCEL");
```

```
alert("User Clicked on: " +MyResult);
```

```
</script>
```

```
</body>
```

Prompt Box

```
<!doctype html>  
<body>  
<script type="text/javascript" language="javascript">  
var MyName=prompt("Enter Your Name:", "KSRaju");  
alert("User Entered: " +MyName);  
</script>  
</body>
```

JavaScript in HTML File

JavaScript Place in HTML File:

There is a flexibility given to include JavaScript code anywhere in HTML document. But there are following most preferred ways to include JavaScript in your HTML file.

1. Script in <head> </head> section
2. Script in <body> </body> section
3. Script in <body> <body> and <head> </head> sections
4. Script in and external file and then include in <head> </head> section

Using an External JavaScript JavaScript can also be placed in external files. External JavaScript files often contain code to be used on several different web pages. External JavaScript files have the file extension js.

Note: External script cannot contain the <script></script> tags!

<head> section contains:

1. <title>
2. <link>
3. <meta>
4. <script>
5. <style>

JavaScript in the <head> Section

```
<!doctype html>
```

```
<head>
```

```
<script type="text/javascript" language="javascript">
```

```
document.write("Hello Welcome to JavaScript<br/>");
```

```
document.write("Hello Welcome to LiveScript");
```

```
</script>
```

```
</head>
```

JavaScript in the <body> Section

```
<!doctype html>
```

```
<body>
```

```
<script type="text/javascript" language="javascript">  
document.write("Hello Welcome to JavaScript<br/>");  
document.write("Hello Welcome to LiveScript");
```

```
</script>
```

```
</body>
```

JavaScript in both <head> and <body> Section

```
<!doctype html>
```

```
<head>
```

```
<script type="text/javascript" language="javascript">
```

```
document.write("Hello Welcome to JavaScript");
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<script type="text/javascript" language="javascript">
```

```
document.write("Hello Welcome to LiveScript");
```

```
</script>
```

```
</body>
```

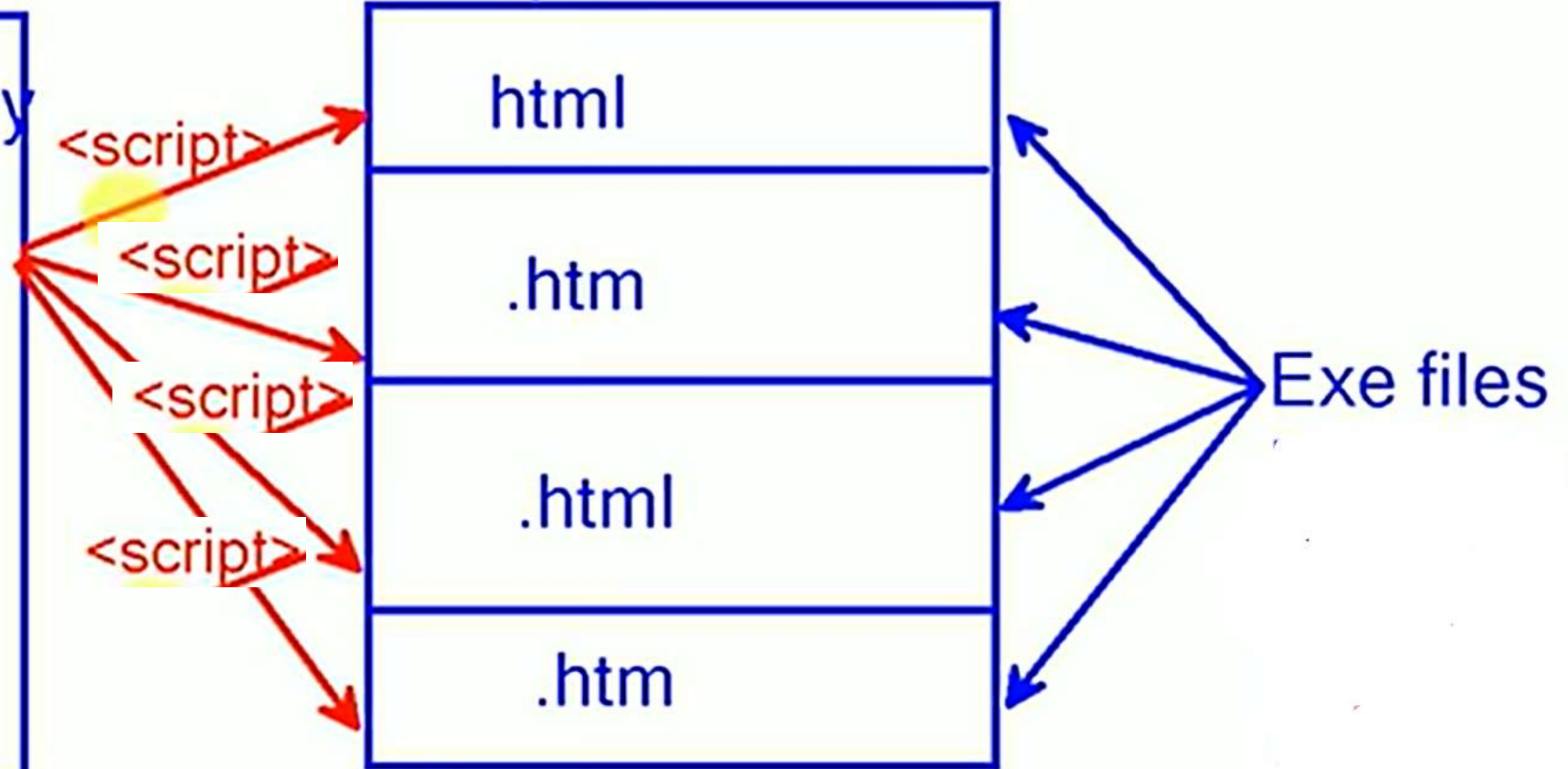
External JavaScripts

Step1

It contains only
JS Code.
It is equal to
normal text
file
`type='text/
javascript'>`

ExternalJS

Step2



Using an External JavaScript

JavaScript can also be placed in external files. External JavaScript files often contain code to be used on several different web pages. External JavaScript files have the file extension .js

Note: External script cannot contain the <script></script> tags!

To use an external script, point to the file in the "src" attribute of the <script> tag

1. Step1-> Create JavaScript File

```
document.write("<h1>Welcome to JS External Programming!!</h1>");  
document.write("<b>Bye");
```

Save this file with .js extension

2. Step2-> Create HTML file

```
<html>  
<head>  
<script src="Example.js"></script>  
</head>  
<body>  
</body>
```

JavaScript Variables

JavaScript Variables

Variable Declaration

- In Programming: **Explicit**
- In Scripting: **Implicit**

Variable use the following naming rules;

1. A variable may include only the letters a-z, A-Z, 0-9, the \$ symbol and the underscore (_).
2. No other characters such as space or punctuation are allowed in a variable name.
3. The first character of a variable name can be only a-z, A-Z, \$, or _ (no numbers).
4. Names are case sensitive. Count, count and COUNT are all different variables.
5. There is no set limit on variable name lengths (but should be reasonable length)
6. Variable name must begin with a letter, the \$ character, or the underscore character.
7. You declare JavaScript variables with the var keyword.

JavaScript Datatypes

JavaScript DataTypes

In JavaScript DataTypes are divided into the following types:

1. Primitive DataTypes
2. Non Primitive DataTypes

1. Primitive DataTypes

- Number (Integer, Float),
- String,
- Boolean,
- Undefined and
- Null.

2. Primitive DataTypes

- Objects

JavaScript DataTypes (contd...)

Number DataTypes

JavaScript has only one type of numbers. Numbers can be written with, or without decimals:

Example:

```
var x1=34.00; // With decimals
```

```
var x2=34;    // Without decimals
```

JavaScript DataTypes (contd...)

String DataTypes

A string is a variable which stores a series of characters like “meet”. A string can be any text inside quotes. You can use single or double quotes.

Example:

```
var name = “meet”
```

```
var name = ‘meet’
```

JavaScript DataTypes (contd...)

```
<!doctype html>
<head>
<title>datatypes and variables</title>
<script>
document.write("<h1>The primitive datatypes</h1>");
var x=20;
document.write("The value of x=" +x + "</br>");
var y=30.67;
document.write("The value of y=" +y + "</br>");
var str='JavaScript String';
document.write("The string is:" +str + "</br>");
document.write("My Boolean is:" +(100>101) + "</br>");
document.write("My Boolean is:" +(101>100) + "</br>");
var z;
document.write("My value is:" +z + "</br>");
var a=null;
document.write("My value is:" +a);
</script>
</head>
<body>
</body>
</html>
```

JavaScript DataTypes (contd...)

Non Primitive DataTypes

JavaScript Arrays

JavaScript arrays are written with square brackets. Array items are separated by commas.

Example

```
<!DOCTYPE html>
<head>
</head>
<body>
<h2>JavaScript Arrays</h2>
<p id="demo"></p>
<script>
const cars = ["Saab","Volvo","BMW"];
document.getElementById("demo").innerHTML = cars[0];
</script>
</body>
</html>
```

JavaScript DataTypes (contd...)

Non Primitive DataTypes

JavaScript Objects

JavaScript objects are written with curly braces {}. Object properties are written as name:value pairs, separated by commas.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
const person = {
    firstName : "John",
    lastName  : "Doe",
    age       : 50,
    eyeColor  : "blue"
};
document.getElementById("demo").innerHTML = person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

Escape Sequences

Escape Sequences

```
<!doctype html>
<head>
<title>Escape Sequence</title>
<script>
document.write("My country's name is \"INDIA\" I love my country");
</script>
</head>
<body>

</body>
</html>
```

<noscript> Tag

<noscript> Tag

If you have to do something important using JavaScript, then you can display a warning message to the user using <noscript> tags. You can add a noscript block immediately after the script block as follows.

HTML <noscript> tag

The <noscript> tag is used to provide an alternate content for users that have disabled scripts in their browser or their browser doesn't support client-side scripting. The <noscript> element will only be displayed if scripts are not supported, or are disabled in the user's browser.

Syntax: <noscript>..... </noscript>

<noscript> Tag (contd...)

Example:

```
<!doctype html>
<head>
<title>noscript tag</title>
<script>
document.write("My country's name is \"INDIA\" I love my country");
</script>
</head>
<body>
<noscript style='color:red'>sorry... your browser is not supporting
javascript. Please enable javascript from tools menu</noscript>
</body>
</html>
```

JavaScript Operators

JavaScript Operators

Arithmetic operators

Arithmetic operators are used to perform mathematics. You can use them for the main four operations (addition, subtraction, multiplication, and division) as well as to find the modulus (the remainder after a division) and to increment or decrement a value.

Operator	Description	Example
+	Addition	$a + 12$
-	Subtraction	$a - 22$
*	Multiplication	$a * 7$
/	Division	$a / 3$
%	Modulus	$a \% 6$
++	Increment	$++a$
--	Decrement	$--a$

Arithmetic operators

```
<!doctype html>
<head>
<title>Operators</title>
<script>
document.write("<h1>Arithmetic Operators</h1>");
var a=100;
var b=200;
document.write("Initially a=" +a + "</br>");
document.write("The value of a= " +(a + 12) + "</br>");
document.write("The value of a=" +(a - 22) + "</br>");
document.write("The value of a=" +(a * 7) + "</br>");
document.write("The value of a=" +(a / 3) + "</br>");
document.write("The value of a=" +(a % 6) + "</br>");
document.write("The value of a=" (++a) + "</br>");
document.write("The value of a=" (--a) + "</br>");
</script>
</head>
<body>

</body>
</html>
```

JavaScript Operators

Contd...

Assignment/Comparison operators

Comparison operators are generally used inside a construct such as an if statement where you need to compare two items.

Operator	Description	Example
==	Is equal to	a == 20
!=	Is not equal to	a != 22
>	Is greater than	a > 7
<	Is less than	a < 3
>=	Is greater than equal to	a >= 6
<=	Is less than equal to	a <= 13

Assignment/Comparison operators

```
<!doctype html>
<head>
<title>Operators</title>
<script>
document.write("<h1>Assignment/Comparison Operators</h1>");
var a=100;
var b=200;
document.write("Initially a=" +a + "</br>");
document.write("The value of a is equal to 20: " +(a == 20) + "</br>");
document.write("The value of a is not equal to 22: " +(a != 22) + "</br>");
document.write("The value of a is greater than 7: " +(a > 7) + "</br>");
document.write("The value of a is less than 3: " +(a < 3) + "</br>");
document.write("The value of a is greater than equal to 6: " +(a >= 6) + "</br>");
document.write("The value of a is less than equal to 13: " +(a <= 13) + "</br>");
</script>
</head>
<body>

</body>
</html>
```

JavaScript Operators

Contd...

Logical operators

Logical operators do not include AND and OR or equivalent to &&, and ||, and there is no xor operator.

Operator	Description	Example
&&	AND	a == 1 && b == 2
	OR	a > 0 a < 100
!	NOT	!(a == b)

Logical operators

```
<!doctype html>
<head>
<title>Operators</title>
<script>
document.write("<h1>Logical Operators</h1>");
var a=100;
var b=200;
document.write("Initially a=" +a + " and b= " +b + "</br>");
document.write("The statement (a == 100 && b == 200) is: " +(a == 100 && b == 200) + "</br>");
document.write("The statement (a == 100 || b == 200) is: " +(a == 100 || b == 200) + "</br>");
document.write("The statement (a > 0 && a < 100) is: " +(a > 0 && a < 100) + "</br>");
document.write("The statement (a > 0 || a < 100) is: " +(a > 0 || a < 100) + "</br>");
document.write("<h1>String Concatenation</h1>");
var str1 = 'Dayanand';
var str2 = 'Yadav';
var str3 = str1 +' '+ str2;
document.write("The string is: " +str3);
</script>
</head>
<body>
</body>
</html>
```

JavaScript Operators

Contd...

String Concatenation

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together. To add two or more string variables together, use the + operator.

Example

```
str1 = "Dayanand";  
str2 = "Yadav";  
str3 = str1 + str2;
```

String Concatenation

```
<!doctype html>
<head>
<title>Operators</title>
<script>
document.write("<h1>String Concatenation</h1>");
var str1 = 'Dayanand';
var str2 = 'Yadav';
var str3 = str1 + ' ' + str2;
document.write("The string is: " +str3);
</script>
</head>
<body>

</body>
</html>
```

JavaScript Conditional Control Statements

JavaScript Conditional Control Statements

Conditional Control Statements

Very often when you write code, you want to perform **different actions for different decisions**. You can use conditional statement in your code to do this.

In JavaScript we have the following conditional statements:

1. **if statement:** use this statement to execute some code only if a specified condition is true.
2. **if else statement:** use this statement to execute some code if the condition is true and another code if the condition is false.
3. **if else if else statement:** use this statement to select one of many blocks of code to be executed.
4. **switch statement:** use this statement to select one of many blocks of code to be executed. switch case statements are alternative ways of executing statements selectively based on certain conditions.

JavaScript Conditional Control Statements

Contd.....

1. **if statement:**

use this statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
{
True statements;
True statements;
True statements;
}
```


JavaScript Conditional Control Statements

Contd.....

2. **if else statement:**

use this statement to execute some code if the condition is true and another code if the condition is false.

Syntax

```
if (condition)
{
True block statement;
}
else
{
False block statement;
}
```

JavaScript Conditional Control Statements

Contd.....

3. **if else if else statement:**

use this statement to select one of many blocks of code to be executed.

Syntax:

```
if(condition)
{
Code to be executed if condition 1 is true
}
else if (condition 2)
{
Code to be executed if condition 2 is true
}
Else
{
Code to be executed if neither condition 1 nor condition 2 is true
}
```

Example

```
<!doctype html>
<head>
<title>If else</title>
<script>
var x = prompt("Enter any number:");
if(x>100)
{
alert("User entered number is greater:" +x);
}
else if(x<100)
{
alert("User entered number is less:" +x);
}
else if(x==100)
{
alert("user entered number is equal: " +x);
}
else
{
alert("Invalid input!")
}
</script>
</head>
<body>

</body>
</html>
```

JavaScript Conditional Control Statements

Contd.....

4. **switch statement:**

use this statement to select one of many blocks of code to be executed. switch case statements are alternative ways of executing statements selectively based on certain conditions.

Syntax

```
switch(n)
```

```
{
```

```
case 1:
```

```
    execute code block 1;
```

```
    break;
```

```
case 2:
```

```
    execute code block 2;
```

```
    break;
```

```
default:
```

```
    code to be executed if n is different from case 1 and 2
```

```
}
```

switch statement Example:

```
<doctype html>
<head>
<title>switch case</title>
<script>
var day = prompt("Enter the name of any day:");
var day_no =parseInt(day);
switch(day_no)
{
case 1:
    document.write( "Monday");
    break;

case 2:
    document.write( "Tuesday");
    break;

case 3 :
    document.write( "Wednesday");
    break;

default:
    document.write("You entered a invalid day");
}
</script>
</head>
<body>

</body>
</html>
```

Looping Control Statement

Looping Control Statement

JavaScript Looping Statements

In JavaScript, there are two different kind of loops:

1. for – loops through a block of code a specified number of times.
2. while – loops through a block of code while a specified condition is true.

for loops

A for loop combines the best of all worlds into a single looping construct that allows you to pass three parameters for each statement:

- A initialization expression
- A condition expression
- A modification expression

These are separated by semicolons, like this:

```
for(exp1; exp2; exp3)
```

Syntax:

```
for(initialization; test condition; iteration statement)
{
Statement(s) to be executed if test condition is true
Statement(s) to be executed if test condition is true
Statement(s) to be executed if test condition is true
}
```

For Loop Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>For loop</title>
  <script>
    for(i=1;i<=10;i++)
    {
      document.write("The value of i= " +i + "</br>");
    }
  </script>
</head>
<body>

</body>
</html>
```


Looping Control Statement (contd.....)

JavaScript while loop

There are two key parts to a JavaScript while loop:

1. The conditional statement which must be True for the while loop's code to be executed.
2. The while loop's code that is contained in curly braces "{and}" will be executed if the condition is True.

Syntax:

```
while(variable<=endvalue)
{
Code to be executed
Code to be executed
}
```

JavaScript do.....while loop

When you require a loop to iterate at least once before any tests are made, use a do.....while loop, which is similar to a while loop, except that the test expression is checked only after each iteration of the loop.

Syntax:

```
do
{
Code to be executed
Code to be executed
}
while(variable<=endvalue);
```

while loop Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>while loop</title>
  <script>
    var i=1;
    while(i<=10)
    {
      document.write("The number is " +i + "</br>");
      i++;
    }
  </script>
</head>
<body>

</body>
</html>
```

do.....while loop Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>do while loop</title>
  <script>
    var i=1;
    do
    {
      document.write("The number is " +i +"</br>");
      i++;
    }
    while(i<=10);
  </script>
</head>
<body>

</body>
</html>
```

JavaScript Functions

JavaScript Functions

What's a Function?

A JavaScript function is a block of code designed to perform a particular task. A function is a block of code that will be executed only by an occurrence of an event at that time function is called. A function can be called from anywhere within the HTML page.

Function can be defined in the beginning of the <head> Tag. JavaScript functions are defined with the function keyword. You can use a function declaration or a function expression.

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing same code again and again. This will help programmers to write modular code. This benefit is also known as "code reusability".

NOTE:

Functions can be defined both in the <head> and in the <body> section of a document

JavaScript Functions contd.....

JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses (). Function names can contain letters, digits, underscores, and dollar signs.

Syntax

```
function functionname (var1, var2,..., varX)
{
Specify Some code
Specify Some code
Specify Some code
}
```

OR

```
function functionName(parameters)
```

JavaScript Function Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JS Functions</title>
  <script>
    function MyMsg()
    {
      document.write("Welcome to the CDGI");
    }
  </script>
</head>
<body>
  <p>Click on button to display the message on browser</p>
  <button onclick="MyMsg()">Click here</button>
</br>
</br>
</body>
</html>
```

Events in JavaScript

Events in JavaScript

Events in JavaScript

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onclick event of a button element to indicate that a function will run when a user clicks on the button. Events are normally used in combination with functions.

Examples of events:

Clicking a button (or any other HTML element)

A page is finished loading

An image is finished loading

Events in JavaScript contd.....

onclick Event

```
<!doctype html>
<head>
<title></title>
<script>
function MySysDate()
{
document.getElementById("dt").innerHTML=Date();
}
</script>
</head>
<body>
<p id="dt">Click on button to display current Date</p>
<button onclick="MySysDate()">Click_me</button>
</body>
</html>
```

Events in JavaScript contd.....

onclick Event

```
<!doctype html>
<head>
<title></title>
<script>
function MySysDate()
{
document.getElementById("dt").innerHTML=Date();
}
</script>
</head>
<body>
<p id="dt">Click on button to display current Date</p>
<button onclick="MySysDate()">Click_me</button>
</body>
</html>
```

Events in JavaScript contd.....

onclick Event

```
<!doctype html>
<head>
<title></title>
<script>
function MySysDate()
{
document.getElementById("dt").innerHTML=Date();
}
</script>
</head>
<body>
<p id="dt">Click on button to display current Date</p>
<button onclick="MySysDate()">Click_me</button>
</body>
</html>
```

// onmouseover & onmouseout Events

```
<!doctype html>
<head>
<title></title>
<script>
function MouseOverBig(obj)
{
obj.style.height="300px";
obj.style.width="300px";
}
function MouseOutSmall(obj)
{
obj.style.height="30px";
obj.style.width="30px";
}
</script>
</head>
<body>

</body>
</html>
```

//onresize event

```
<!doctype html>
<head>
<title>
</title>
<script>
function PageResize()
{
    alert("Sorry Page Resize no allowd");
}
</script>
</head>
<body onresize="PageResize()">
<p>Sorry Page Resize no allowd</p>
</body>
</html>
```

//onselect Event

```
<head>
<title>
</title>
<script>
function TextSelect()
{
    alert("Sorry you can not select text from form element");
}
</script>
</head>
<body>
<form>
<label>Enter user name</label><br/>
<input type="text" name="fname" onselect="TextSelect()"> <br/>
<label>Enter Password</label><br/>
<input type="password" name="pwd"><br/>
</form>
</body>
</html>
```

//onblur Event

onblur event occurs when an object loses focus, onblur is most often used with form validation code (when a user leaves a form field).

Note: the onblur event is the opposite of the onfocus event.

Syntax:

```
<element onblur="someJavaScript code">
```


Events in JavaScript contd.....

```
<!doctype html>
<head>
<title>
</title>
<script>
function UpperCase()
{
var x = document.getElementById("fname");
x.value = x.value.toUpperCase();

var x = document.getElementById("lname");
x.value = x.value.toUpperCase();

}
</script>
</head>
<body>
<form>
<label>Enter your first name (onblur)</label><br/>
<input type="text" id="fname" onblur="UpperCase()"><br/>
<label>Enter your last name (onfocus)</label><br/>
<input type="text" id="lname" onfocus="UpperCase()"><br/>
</form>
</body>
</html>
```

Global Functions in JavaScript

Global Functions in JavaScript

parseInt and parseFloat:

To convert a string to number, use the JavaScript functions

1. parseInt (for string to integer conversion)
2. parseFloat (For conversion to a floating point number)

Syntax

parseInt(string)

```
<!doctype html>
<head>
<title></title>
<script>
var x = "100";
var y = 200;
var z = parseInt(x)+y;
document.write("The sum is = " +z);
</script>
</head>
<body>
</body>
</html>
```

Global Functions in JavaScript

parseInt and parseFloat:

To convert a string to number, use the JavaScript functions

1. parseInt (for string to integer conversion)
2. parseFloat (For conversion to a floating point number)

Syntax

parseInt(string)

```
<!doctype html>
<head>
<title></title>
<script>
var x = prompt("Enter any number");
var y = prompt("Enter any number");
//var z = x-y;
var z = parseInt(x)+parseInt(y);
document.write("The sum is = " +z);
</script>
</head>
<body>
</body>
</html>
```

Global Functions in JavaScript

parseFloat:

parseFloat (For conversion to a floating point number)

Syntax

parseFloat(string)

```
<!doctype html>
<head>
<title></title>
<script>
var x = "10.2";
var y = "5.7";
var z = parseFloat(x)+parseFloat(y);
document.write("The sum is = " +z);
</script>
</head>
<body>
</body>
</html>
```

Global Functions in JavaScript

parseFloat:

parseFloat (For conversion to a floating point number)

Syntax

parseFloat(string)

```
<!doctype html>
<head>
<title></title>
<script>
var x = prompt("Enter any number");
var y = prompt("Enter any number");
//var z = x-y;
var z = parseFloat(x)+parseFloat(y);
document.write("The sum is = " +z);
</script>
</head>
<body>
</body>
</html>
```

Errors in JavaScript

Errors in JavaScript

Errors and Exception Handling


There are three types of errors in programming

- (a) Syntax Errors
- (b) Runtime Errors
- (c) Logical Errors

Syntax Errors

Syntax errors are also called parsing errors, occurs at interpret time in JavaScript (compile time in traditional programming languages).

```
<!doctype html>  
<head>  
<script>  
document.write("Welcome to JavaScript"  
</script>  
</head>  
</html>
```



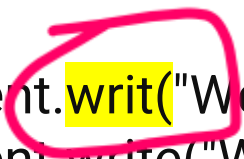
Above example causes syntax error because it is missing a closing parenthesis.

Errors in JavaScript

Runtime Errors

Runtime errors occurs during execution.

```
<!doctype html>  
<head>  
<script>  
document.writ("Welcome to JavaScript");  
document.write("Welcome to LiveScript");  
</script>  
</head>  
</html>
```



Above example causes runtime error because here syntax is correct but at runtime it is trying to call a non existed method.

Errors in JavaScript

The try.....catch.....finally

The try.....catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an errors occurs.

Here is the try.....catch.....finally block

Syntax

```
<script>
Try
{
Code to run[break;]
}
Catch (e)
{
Code to run if an exception occurs [break;]
}
finally
{
Code that is always executed regardless of // an exception occuring
}
</script>
```

Errors in JavaScript

The try.....catch.....finally

Example1

```
<!doctype html>
<head>
<script>
try
{
aler("Welcome to LiveScript");
}
catch(e)
{
alert(e.description);
}
alert("welcome to JavaScript");
</script>
</head>
</html>
```

Errors in JavaScript

The try.....catch.....finally

Example2

```
<!doctype html>
<head>
<script>
var x = prompt("Enter any number" );
try
{
alert(eval(x));
}
catch(e)
{
alert("only numeric value required" +e.description);
}
alert("Successfully Executed");
</script>
</head>
</html>
```

Objects in JavaScript

Objects in JavaScript

JavaScript Objects

Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to an **object** named car:

```
const car = {type:"Fiat", model:"500", color:"white"};
```

Syntax (Array)

```
var Array1 = [values];
```

Or

```
var Array1 = new Array (values);
```

Or

```
var Array1 = new Array;
```

```
Array[0] = value;
```

```
Array[1] = value;
```

```
.
```

```
.
```

```
Array[n] = value;
```

Objects in JavaScript

Example

```
<!DOCTYPE html>
<head>
<script>
// Create an Object:
var car = {type:"Fiat", model:"F500", color:"white"};
function CarType()
{
// Display Data from the Object:
document.getElementById("type").innerHTML = car.type;
}
function CarModel()
{
// Display Data from the Object:
document.getElementById("type").innerHTML = car.model;
}
function CarColor()
{
// Display Data from the Object:
document.getElementById("type").innerHTML = car.color;
}
</script>
</head>
<body>
<h1>JavaScript Objects</h1>
<h2>Creating an Object</h2>

<p> Click on below buttons for car specification:</p>
<span id="type" style='color:red;font-size:50px'></span><br/>
<button onclick="CarType()">Click type</button>
<button onclick="CarModel()">Click model</button>
<button onclick="CarColor()">Click color</button>
</body>
```

Date Objects in JavaScript

JavaScript Date object

Date object is a datatype built into JavaScript Language.

Date Object created with the new Date().

Once a new Date() object is created , a number of methods allow you to operation on it. Most of methods get and set the year, month, day, hour, minute, second and millisecond field on the object, using either local time or UT (Universal Time or GMT) time.

JavaScript Date and Time Object

The Date object is useful when you want to display a date or use a timestamp in some sort of calculation.

JavaScript creates a Date object based on your visitor's system clock.

There are four ways:

```
var x = new Date();
```

```
var x = new Date(milliseconds);
```

```
var x = new Date(dateString);
```

```
var x = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```


Date Objects in JavaScript

JavaScript Date object

1. **getTime()** – number of milliseconds
2. **getSeconds()** – number of seconds (0-59)
3. **getMinutes()** – number of minutes (0-59)
4. **getHours()** – number of hours (0-23)
5. **getDay()** – number of days (0-6) (0-Sunday, 6-Saturday)
6. **getDate()** – number of dates in a month (0-30)
7. **getMonth()** – Number of month (0-11)
8. **getFullYear()** – The four digit year (1970 – 9999)

Date Objects in JavaScript

JavaScript Date object

```
<!DOCTYPE html>
<head>
</head>
<body>
<h1>JavaScript Date Objects</h1>
<script>
var dt = new Date();
var dd = dt.getDate();
var mm = dt.getMonth()+1;
var yyyy = dt.getFullYear();
document.write("The current Date is: " +dd+"/"+mm+"/"+yyyy);
</script>
</body>
</html>
```

Time Objects in JavaScript

JavaScript time object

```
<!DOCTYPE html>
<head>
</head>
<body>
<h1>JavaScript time Objects</h1>
<div id="txt"></div>
<script>
var time = new Date();
var hh = time.getHours();
var min = time.getMinutes();
var ss = time.getSeconds();
document.getElementById('txt').innerHTML="Current Time: " +hh + ":" +min + ":" +ss;
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<head>
</head>
<body onload="currentTime()">

<h1>JavaScript time objects</h1>

<div id="txt"></div>

<script>
function currentTime()
    {
    var time = new Date();
    var hh = time.getHours();
    var min = time.getMinutes();
    var ss = time.getSeconds();
    document.getElementById('txt').innerHTML = "Current Time: " +hh + ":" +min + ":" +ss;
    setTimeout(currentTime, 1000);
    }
</script>

</body>
</html>
```

String Object in JavaScript

JavaScript String Object

A JavaScript string object stores a series of characters like “India”, “JS” etc. A string can be any text inside ‘single’ or “double” quote.

Example

```
var name = “India”
```

```
var name = ‘India’
```

Syntax

```
var str = new String(“India”);
```

Or

```
var str = “India”;
```

String Object in JavaScript

JavaScript String Object Example1 (To find string length & string Index)

```
<!doctype html>
<head>
<script>
var str1="Dayanand";
var str2="Yadav";
document.write("Str1 = " +str1 + "</br>");
document.write("No. of characters in string 1 is : " +str1.length + "</br>");
document.write("Character at index [0] : " +str1.charAt(0) + "</br>");
document.write("=====" + "</br>");
document.write("Str2 = " +str2 + "</br>");
document.write("No. of characters in string 2 is : " +str2.length + "</br>");
document.write("Character at index [4] : : " +str2.charAt(4) + "</br>");
</script>
</head>
<body>
</body>
</html>
```

Document Object

JS Document Object

Each HTML page loaded in to browser window becomes document object. The document object provides the access to all HTML elements in a page, from within a script.

Note: The Document object is also part of the window object, and can be accessed through the window document property.

Document Object

JS Document Object Example

```
<!doctype html>
<head>
<title> Document Object </title>
</head>
<body>



<p id = "demo" style='color:Blue';></p>
<script>
document.getElementById("demo").innerHTML = "Number of images: " + document.images.length;
</script>
</body>
</html>
```


Document Object

JS Document Object Example

```
<!DOCTYPE html>
<html>
<body>
<h2>Finding HTML Elements Using document.anchors</h2>
<a href="html" name="html">HTML</a><br>
<a href="css" name="css">CSS</a><br>
<a href="JavaScript" name="js">JavaScript</a><br>
<p id="demo2"></p>
<script>
document.getElementById("demo2").innerHTML =
"Number of anchors are: " + document.anchors.length;
</script>
</body>
</html>
```

Forms and validations

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
    let x = document.forms["myForm"]["fname"].value;
    if (x == "")
    {
        document.getElementById('alert').innerHTML="* Name must be filled out";
        return false;
    }
}
</script>
</head>
<body>
<h2>JavaScript Validation</h2>
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()"
method="post">
    Name: <input type="text" name="fname">
    <input type="submit" value="Submit"><br>
<span id="alert" style="color: red;"></span>
</form>
</body>
</html>
```

Introduction to CSS



Cascading **S**tyl**e** **S**heets

Introduction to CSS

What is CSS?

- ❑ CSS stands for Cascading Style Sheets.
- ❑ CSS is the language we use to style an HTML document.
- ❑ CSS describes how HTML elements should be displayed.
- ❑ Used to override the default styles of HTML
- ❑ Inline / Block (Internal) / File Modes (External)

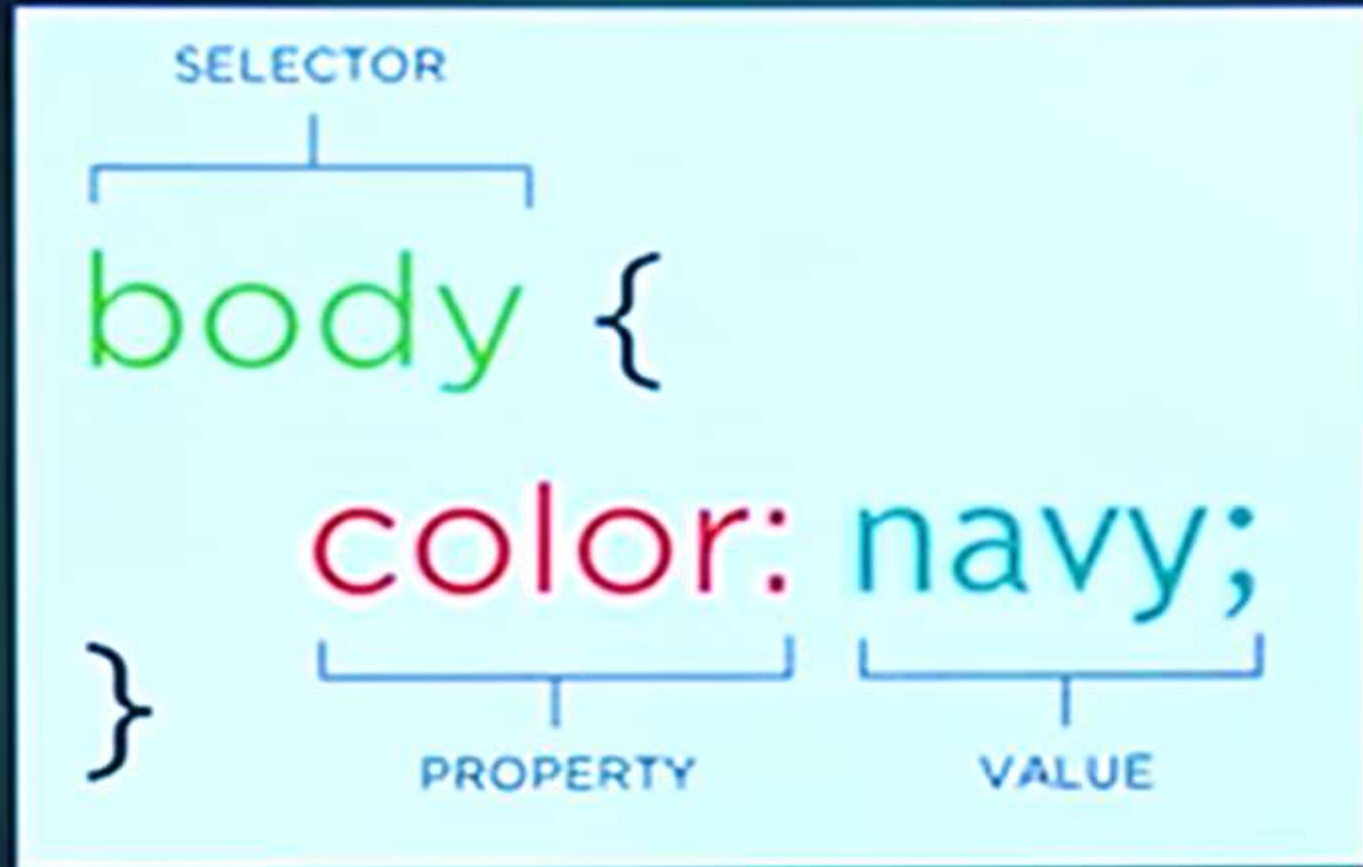
Developed by Håkon Wium Lie and Bert Bos in 1994

HTML + CSS

HTML



CSS Syntax



CSS Selectors

- Element Selector `h1(...)`
- Id selector `#yourId(..)`
- Class selector `.yourClass(...)`
- Attribute Selector `input[type="text"]`
- Pseudo Selector `button:hover`

CSS Selectors

```
<style>
  h1{
    background-color: red;
    color: white;
  }
  p{
    color: navy;
  }
  #p3{
    color: red;
  }
  .p5{
    color: #ff9900;
  }

  input[type="text"]{
    background-color: red;
    color: white;
  }
  input[type="submit"]:hover{
    background-color: green;
    color: white;
  }
</style>
```

Background images

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("img/nature2.jpg");
    background-color: #cccccc;
    background-size: cover;
}
</style>
</head>
<body>

</body>
</html>
```

Colours and properties

```
body{  
    background-color: hsl(10,80%,100%);  
}  
  
h1{  
    color: rgb(255, 100, 50);  
}  
  
#id1{  
    color: #ff9922;  
}
```

Manipulating texts

```
<style>
div {
  border: 1px solid gray;
  padding: 8px;
}
h1 {
  text-align: center;
  text-transform: uppercase;
  color: #4CAF50;
}
p {
  text-indent: 50px;
  text-align: justify;
  letter-spacing: 3px;
}
a {
  text-decoration: none;
  color: #008CBA;
}
</style>
<div>
  <h1>text formatting</h1>
  <p>This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform
  <a target="_blank" href="tryit.asp?filename=trycss_text">"Try it Yourself"</a> link.</p>
</div>
```

Borders and Boxes

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    background-color: lightgrey;
    width: 300px;
    border: 15px solid green;
    padding: 50px;
    margin: 20px;
}
</style>
</head>
<body>
<h2>Borders and Boxes</h2>
<p>CS 504 (A) IWT</p>
<div>UNIT 03
    Style sheets : Need for CSS, introduction to CSS, basic syntax and structure, using CSS,
    background images, colors and properties, manipulating texts, using fonts, borders and boxes,
    margins, padding lists, positioning using CSS, CSS2, Overview and features of CSS3
    JavaScript : Client side scripting with JavaScript, variables, functions, conditions, loops and
    repetition, Pop up boxes, Advance JavaScript: Javascript and objects, JavaScript own objects,
    the DOM and web browser environments, Manipulation using DOM, forms and
    validations, DHTML : Combining HTML, CSS and Javascript, Events and buttons
</div>
</body>
</html>
```

Margin

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 2px solid black;
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  background-color: lightblue;
}
</style>
</head>
<body>
```

```
<h2>Margin</h2>
```

```
<div>top margin of 100px, a right margin of 150px, a bottom margin
of 100px, and a left margin of 80px.</div>
```

```
</body>
</html>
```

Padding

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  background-color: lightblue;
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
</style>
</head>
<body>
```

```
<h2>Padding</h2>
```

```
<div>top padding of 50px, a right padding of 30px, a bottom
padding of 50px, and a left padding of 80px.</div>
```

```
</body>
</html>
```

Position: static

```
<head>
<style>
div.static {
  position: static;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
```

```
<h2>Position: static</h2>
```

<p>An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:</p>

```
<div class="static">
This div element has position: static;
</div>
```

```
</body>
</html>
```


Position: relative

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 60px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
```

```
<h2>position: relative;</h2>
```

<p>An element with position: relative; is positioned relative to its normal position:</p>

```
<div class="relative">
This div element has position: relative;
</div>
```

```
</body>
</html>
```

position: fixed

```
<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
```

```
<h2>position: fixed;</h2>
```

<p>An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:</p>

```
<div class="fixed">
```

This div element has position: fixed;

```
</div>
```

```
</body>
```

```
</html>
```

position: absolute

```
<!DOCTYPE html>
<html>
<head>
<style>
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;
}
</style>
</head>
<body>
<h2>position: absolute;</h2>
<p>An element with position: absolute; is positioned relative to the nearest
positioned ancestor (instead of positioned relative to the viewport, like
fixed):</p>
<div class="relative">This div element has position: relative;
    <div class="absolute">This div element has position: absolute;</div>
</div>
</body>
</html>
```

Position:
sticky

```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
position: sticky; top: 0; padding: 5px; background-color: #cae8ca; border: 2px solid
#4CAF50;
}
</style>
</head>
<body>
```

<p>Try to scroll inside this frame to understand how sticky positioning works.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">

<p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>

<p>Scroll back up to remove the stickiness.</p>

<p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisncludaturque et eum, </p>

<p>Some text to enable scrolling.. Lorem ipsum dolor sit omnis.</p>

</div>

</body>

</html>

All CSS Versions with Features

CSS1:

- Introduced basic styling properties for fonts, text, colors, and backgrounds.
- Introduced the concept of selectors for specifying which elements to style.
- Provided support for font properties, text alignment, and background properties.

All CSS Versions with Features

CSS2:

- Expanded layout capabilities with positioning and floating elements.
- Introduced the display property for controlling how elements are rendered.
- Introduced pseudo-classes and pseudo-elements for more specific targeting.
- Added support for media types (e.g., screen, print) and handling print styles.
- Improved support for internationalization, including bidirectional text.

All CSS Versions with Features

CSS2.1:

- Refined the CSS2 specification to address inconsistencies and ambiguities.
- Standardized box model properties, improving cross-browser compatibility.
- Introduced CSS3 features like color properties (e.g., `rgba()`, `hsl()`), but they were not widely implemented.

CSS3:

Selectors: Introduced advanced selectors, such as attribute selectors and the `:nth-child` pseudo-class.

Box Model: Enhanced the box model with properties like `box-sizing`, allowing control over how box dimensions are calculated.

Typography: Added features for web fonts (e.g., `@font-face`) and text shadow.

Backgrounds and Borders: Introduced gradients, multiple background images, and rounded corners with `border-radius`.

Colors: Introduced new color formats (e.g., `rgba()`, `hsl()`) and properties like `opacity`.

2D Transforms: Provided transformation properties like `rotate`, `scale`, `translate`, and `skew` for elements.

Transitions and Animations: Introduced the ability to create smooth transitions and animations between CSS property values.

Flexible Box Layout (Flexbox): A layout module for creating flexible and responsive layouts.

Grid Layout: Introduced grid-based layout with the `display: grid` property.

Multi-Column Layout: Supported multi-column text layout.

Media Queries: Enabled responsive design by allowing styles to adapt based on the device's characteristics (e.g., screen size).

Responsive Web Design: Popularized responsive design principles, encouraging adaptable layouts for various devices.

Custom Properties (CSS Variables): Introduced custom properties to store and reuse values within CSS.

Thank You