**SILVER OAK UNIVERSIRTY**

**College Of Computer Application**

**Enrollment No : 2204030102703**

**Name :Kirtan Thakar**

**DAY -3 TASK**

**1.) HOVER ELEMENT**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**a:hover {**

**background-color: yellow;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>Demo of the :hover selector</h1>**

**<p>The :hover selector style links on mouse-over:</p>**

**<a href="https://www.GOOGLE.com">GOOGLE.com</a>**

**<a href="https://www.wikipedia.org">wikipedia.org</a>**

**</body>**

**</html>**

**OUTPUT :**



**\*Mark up languages**

**2)XML - Extensible Markup Language.**



**3) SGML**

**SGML (standard generalized markup language) is an international standard for the definition of markup languages; that is, it is a metalanguage. Markup consists of notations called tags that specify the function of a piece of text or how it is to be displayed. SGML emphasizes descriptive markup, in which a tag might be “<emphasis>.” Such a markup denotes the document function, and it could be interpreted as reverse video on a computer screen, underlining by a typewriter, or italics in typeset text.**

**SGML is used to specify DTDs (document type definitions). A DTD defines a kind of document, such as a report, by specifying what elements must appear in the document—e.g., <Title>—and giving rules for the use of document elements, such as that a paragraph may appear within a table entry but a table may not appear within a paragraph. A marked-up text may be analyzed by a parsing program to determine if it conforms to a DTD. Another program may read the markups to prepare an index or to translate the document into PostScript for printing. Yet another might generate large type or audio for readers with visual or hearing disabilities.**

**World Wide Web display languages**

**HTML**

**The World Wide Web is a system for displaying text, graphics, and audio retrieved over the Internet on a computer monitor. Each retrieval unit is known as a Web page, and such pages frequently contain “links” that allow related pages to be retrieved. HTML (hypertext markup language) is the markup language for encoding Web pages. It was designed by Tim Berners-Lee at the CERN nuclear physics laboratory in Switzerland during the 1980s and is defined by an SGML DTD. HTML markup tags specify document elements such as headings, paragraphs, and tables. They mark up a document for display by a computer program known as a Web browser. The browser interprets the tags, displaying the headings, paragraphs, and tables in a layout that is adapted to the screen size and fonts available to it.**

**HTML documents also contain anchors, which are tags that specify links to other Web pages. An anchor has the form <A HREF= “**[**http://www.britannica.com**](http://www.britannica.com/)**”> Encyclopædia Britannica</A>, where the quoted string is the URL (uniform resource locator) to which the link points (the Web “address”) and the text following it is what appears in a Web browser, underlined to show that it is a link to another page. What is displayed as a single page may also be formed from multiple URLs, some containing text and others graphics.**

**More From Britannica computer science: Programming languages**

**XML**

**HTML does not allow one to define new text elements; that is, it is not extensible. XML (extensible markup language) is a simplified form of SGML intended for documents that are published on the Web. Like SGML, XML uses DTDs to define document types and the meanings of tags used in them. XML adopts conventions that make it easy to parse, such as that document entities are marked by both a beginning and an ending tag, such as <BEGIN>…</BEGIN>. XML provides more kinds of hypertext links than HTML, such as bidirectional links and links relative to a document subsection.**

**Because an author may define new tags, an XML DTD must also contain rules that instruct a Web browser how to interpret them—how an entity is to be displayed or how it is to generate an action such as preparing an e-mail message.**

**Web scripting**

**Web pages marked up with HTML or XML are largely static documents. Web scripting can add information to a page as a reader uses it or let the reader enter information that may, for example, be passed on to the order department of an online business. CGI (common gateway interface) provides one mechanism; it transmits requests and responses between the reader’s Web browser and the Web server that provides the page. The CGI component on the server contains small programs called scripts that take information from the browser system or provide it for display. A simple script might ask the reader’s name, determine the Internet address of the system that the reader uses, and print a greeting. Scripts may be written in any programming language, but, because they are generally simple text-processing routines, scripting languages like PERL are particularly appropriate.**

**Another approach is to use a language designed for Web scripts to be executed by the browser. JavaScript is one such language, designed by the Netscape Communications Corp., which may be used with both Netscape’s and Microsoft’s browsers. JavaScript is a simple language, quite different from Java. A JavaScript program may be embedded in a Web page with the HTML tag <script language=“JavaScript”>. JavaScript instructions following that tag will be executed by the browser when the page is selected. In order to speed up display of dynamic (interactive) pages, JavaScript is often combined with XML or some other language for exchanging information between the server and the client’s browser. In particular, the XMLHttpRequest command enables asynchronous data requests from the server without requiring the server to resend the entire Web page. This approach, or “philosophy,” of programming is called Ajax (asynchronous JavaScript and XML).**

**VB Script is a subset of Visual Basic. Originally developed for Microsoft’s Office suite of programs, it was later used for Web scripting as well. Its capabilities are similar to those of JavaScript, and it may be embedded in HTML in the same fashion.**

**Behind the use of such scripting languages for Web programming lies the idea of component programming, in which programs are constructed by combining independent previously written components without any further language processing. JavaScript and VB Script programs were designed as components that may be attached to Web browsers to control how they display information.**

**Elements of programming**

**Despite notational differences, contemporary computer languages provide many of the same programming structures. These include basic control structures and data structures. The former provide the means to express algorithms, and the latter provide ways to organize information.**

**Control structures**

**Programs written in procedural languages, the most common kind, are like recipes, having lists of ingredients and step-by-step instructions for using them. The three basic control structures in virtually every procedural language are:**

**1. Sequence—combine the liquid ingredients, and next add the dry ones.**

**2. Conditional—if the tomatoes are fresh then simmer them, but if canned, skip this step.**

**3. Iterative—beat the egg whites until they form soft peaks.**

**Sequence is the default control structure; instructions are executed one after another. They might, for example, carry out a series of arithmetic operations, assigning results to variables, to find the roots of a quadratic equation ax2 + bx + c = 0. The conditional IF-THEN or IF-THEN-ELSE control structure allows a program to follow alternative paths of execution. Iteration, or looping, gives computers much of their power. They can repeat a sequence of steps as often as necessary, and appropriate repetitions of quite simple steps can solve complex problems.**

**These control structures can be combined. A sequence may contain several loops; a loop may contain a loop nested within it, or the two branches of a conditional may each contain sequences with loops and more conditionals. In the “pseudocode” used in this article, “\*” indicates multiplication and “←” is used to assign values to variables. The following programming fragment employs the IF-THEN structure for finding one root of the quadratic equation, using the quadratic formula:**

**quadratic formula.**

**The quadratic formula assumes that a is nonzero and that the discriminant (the portion within the square root sign) is not negative (in order to obtain a real number root). Conditionals check those assumptions:**

**IF a = 0 THEN**

**ROOT ← −c/b**

**ELSE**

**DISCRIMINANT ← b\*b − 4\*a\*c**

**IF DISCRIMINANT ≥ 0 THEN**

**ROOT ← (−b + SQUARE\_ROOT(DISCRIMINANT))/2\*a**

**ENDIF**

**ENDIF**

**The SQUARE\_ROOT function used in the above fragment is an example of a subprogram (also called a procedure, subroutine, or function). A subprogram is like a sauce recipe given once and used as part of many other recipes. Subprograms take inputs (the quantity needed) and produce results (the sauce). Commonly used subprograms are generally in a collection or library provided with a language. Subprograms may call other subprograms in their definitions, as shown by the following routine (where ABS is the absolute-value function). SQUARE\_ROOT is implemented by using a WHILE (indefinite) loop that produces a good approximation for the square root of real numbers unless x is very small or very large. A subprogram is written by declaring its name, the type of input data, and the output:**

**FUNCTION SQUARE\_ROOT(REAL x) RETURNS REAL**

**ROOT ← 1.0**

**WHILE ABS(ROOT\*ROOT − x) ≥ 0.000001**

**AND WHILE ROOT ← (x/ROOT + ROOT)/2**

**RETURN ROOT**

**Subprograms can break a problem into smaller, more tractable subproblems. Sometimes a problem may be solved by reducing it to a subproblem that is a smaller version of the original. In that case the routine is known as a recursive subprogram because it solves the problem by repeatedly calling itself. For example, the factorial function in mathematics (n! = n∙(n−1)⋯3∙2∙1—i.e., the product of the first n integers), can be programmed as a recursive routine:**

**FUNCTION FACTORIAL(INTEGER n) RETURNS INTEGER**

**IF n = 0 THEN RETURN 1**

**ELSE RETURN n \* FACTORIAL(n−1)**

**The advantage of recursion is that it is often a simple restatement of a precise definition, one that avoids the bookkeeping details of an iterative solution.**

**At the machine-language level, loops and conditionals are implemented with branch instructions that say “jump to” a new point in the program. The “goto” statement in higher-level languages expresses the same operation but is rarely used because it makes it difficult for humans to follow the “flow” of a program. Some languages, such as Java and Ada, do not allow it.**

**Data structures**

**Whereas control structures organize algorithms, data structures organize information. In particular, data structures specify types of data, and thus which operations can be performed on them, while eliminating the need for a programmer to keep track of memory addresses. Simple data structures include integers, real numbers, Booleans (true/false), and characters or character strings. Compound data structures are formed by combining one or more data types.**

**The most important compound data structures are the array, a homogeneous collection of data, and the record, a heterogeneous collection. An array may represent a vector of numbers, a list of strings, or a collection of vectors (an array of arrays, or mathematical matrix). A record might store employee information—name, title, and salary. An array of records, such as a table of employees, is a collection of elements, each of which is heterogeneous. Conversely, a record might contain a vector—i.e., an array.**

**Record components, or fields, are selected by name; for example, E.SALARY might represent the salary field of record E. An array element is selected by its position or index; A[10] is the element at position 10 in array A. A FOR loop (definite iteration) can thus run through an array with index limits (FIRST TO LAST in the following example) in order to sum its elements:**

**FOR i ← FIRST TO LAST**

**SUM ← SUM + A[i]**

**Arrays and records have fixed sizes. Structures that can grow are built with dynamic allocation, which provides new storage as required. These data structures have components, each containing data and references to further components (in machine terms, their addresses). Such self-referential structures have recursive definitions. A bintree (binary tree) for example, either is empty or contains a root component with data and left and right bintree “children.” Such bintrees implement tables of information efficiently. Subroutines to operate on them are naturally recursive; the following routine prints out all the elements of a bintree (each is the root of some subtree):**

**PROCEDURE TRAVERSE(ROOT: BINTREE)**

**IF NOT(EMPTY(ROOT))**

**TRAVERSE(ROOT.LEFT)**

**PRINT ROOT.DATA**

**TRAVERSE(ROOT.RIGHT)**

**ENDIF**

**Abstract data types (ADTs) are important for large-scale programming. They package data structures and operations on them, hiding internal details. For example, an ADT table provides insertion and lookup operations to users while keeping the underlying structure, whether an array, list, or binary tree, invisible. In object-oriented languages, classes are ADTs and objects are instances of them. The following object-oriented pseudocode example assumes that there is an ADT bintree and a “superclass” COMPARABLE, characterizing data for which there is a comparison operation (such as “<” for integers). It defines a new ADT, TABLE, that hides its data-representation and provides operations appropriate to tables. This class is polymorphic—defined in terms of an element-type parameter of the COMPARABLE class. Any instance of it must specify that type, here a class with employee data (the COMPARABLE declaration means that PERS\_REC must provide a comparison operation to sort records). Implementation details are omitted.**

**CLASS TABLE OF <COMPARABLE T>**

**PRIVATE DATA: BINTREE OF <T>**

**PUBLIC INSERT(ITEM: T)**

**PUBLIC LOOKUP(ITEM: T) RETURNS BOOLEAN**

**END**

**CLASS PERS\_REC: COMPARABLE**

**PRIVATE NAME: STRING**

**PRIVATE POSITION: {STAFF, SUPERVISOR, MANAGER}**

**PRIVATE SALARY: REAL**

**PUBLIC COMPARE (R: PERS\_REC) RETURNS BOOLEAN**

**END**

**EMPLOYEES: TABLE <PERS\_REC>**

**TABLE makes public only its own operations; thus, if it is modified to use an array or list rather than a bintree, programs that use it cannot detect the change. This information hiding is essential to managing complexity in large programs. It divides them into small parts, with “contracts” between the parts; here the TABLE class contracts to provide lookup and insertion operations, and its users contract to use only the operations so publicized.**

**HTML: HyperText Markup Language**

**HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript).**

**"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.**

**HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li> and many others.**

**An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the <title> tag can be written as <Title>, <TITLE>, or in any other way. However, the convention and recommended practice is to write tags in lowercase.**

**The articles below can help you learn more about HTML.**

**Key resources**

**HTML Introduction**

**If you're new to web development, be sure to read our HTML Basics article to learn what HTML is and how to use it.**

**HTML Tutorials**

**For articles about how to use HTML, as well as tutorials and complete examples, check out our HTML Learning Area.**

**HTML Reference**

**In our extensive HTML reference section, you'll find the details about every element and attribute in HTML.**

**Looking to become a front-end web developer?**

**We have put together a course that includes all the essential information you need to work towards your goal.**

**Get started**

**Beginner's tutorials**

**Our HTML Learning Area features multiple modules that teach HTML from the ground up — no previous knowledge required.**

**Introduction to HTML**

**This module sets the stage, getting you used to important concepts and syntax such as looking at applying HTML to text, how to create hyperlinks, and how to use HTML to structure a web page.**

**Multimedia and embedding**

**This module explores how to use HTML to include multimedia in your web pages, including the different ways that images can be included, and how to embed video, audio, and even entire other webpages.**

**HTML tables**

**Representing tabular data on a webpage in an understandable, accessible way can be a challenge. This module covers basic table markup, along with more complex features such as implementing captions and summaries.**

**HTML forms**

**Forms are a very important part of the Web — these provide much of the functionality you need for interacting with websites, e.g. registering and logging in, sending feedback, buying products, and more. This module gets you started with creating the client-side/front-end parts of forms.**

**Use HTML to solve common problems**

**Provides links to sections of content explaining how to use HTML to solve very common problems when creating a web page: dealing with titles, adding images or videos, emphasizing content, creating a basic form, etc.**

**Advanced topics**

**CORS enabled image**

**The crossorigin attribute, in combination with an appropriate CORS header, allows images defined by the <img> element to be loaded from foreign origins and used in a <canvas> element as if they were being loaded from the current origin.**

**CORS settings attributes**

**Some HTML elements that provide support for CORS, such as <img> or <video>, have a crossorigin attribute (crossOrigin property), which lets you configure the CORS requests for the element's fetched data.**

**Preloading content with rel="preload"**

**The preload value of the <link> element's rel attribute allows you to write declarative fetch requests in your HTML <head>, specifying resources that your pages will need very soon after loading, which you therefore want to start preloading early in the lifecycle of a page load, before the browser's main rendering machinery kicks in. This ensures that they are made available earlier and are less likely to block the page's first render, leading to performance improvements. This article provides a basic guide to how preload works.**

**Reference**

**HTML reference**

**HTML consists of elements, each of which may be modified by some number of attributes. HTML documents are connected to each other with links.**

**HTML element reference**

**Browse a list of all HTML elements.**

**HTML attribute reference**

**Elements in HTML have attributes. These are additional values that configure the elements or adjust their behavior in various ways.**

**Global attributes**

**Global attributes may be specified on all HTML elements, even those not specified in the standard. This means that any non-standard elements must still permit these attributes, even though those elements make the document HTML5-noncompliant.**

**Inline elements and block-level elements**

**HTML elements are usually "inline" or "block-level" elements. An inline element occupies only the space bounded by the tags that define it. A block-level element occupies the entire space of its parent element (container), thereby creating a "block".**

**Guide to media types and formats on the web**

**The <audio> and <video> elements allow you to play audio and video media natively within your content without the need for external software support.**

**HTML content categories**

**HTML is comprised of several kinds of content, each of which is allowed to be used in certain contexts and is disallowed in others. Similarly, each context has a set of other content categories it can contain and elements that can or can't be used in them. This is a guide to these categories.**

**Quirks mode and standards mode**

**Historical information on quirks mode and standards mode**

**5.)XHTML - Extensible Hypertext Markup Language.**

**What is XHTML**

**XHTML stands for EXtensible HyperText Markup Language. It is a cross between HTML and XML language.**

**XHTML is almost identical to HTML but it is stricter than HTML. XHTML is HTML defined as an XML application. It is supported by all major browsers.**

**Although XHTML is almost the same as HTML but It is more important to create your code correctly, because XHTML is stricter than HTML in syntax and case sensitivity. XHTML documents are well-formed and parsed using standard XML parsers, unlike HTML, which requires a lenient HTML-specific parser.**

**History**

**XHTML 1.0 became a World Wide Web Consortium (W3C) Recommendation on January 26, 2000. XHTML 1.1 became a W3C Recommendation on May 31, 2001. The standard known as XHTML5 is being developed as an XML adaptation of the HTML5 specification.**

**Why use XHTML**

**XHTML was developed to make HTML more extensible and increase interoperability with other data formats. There are two main reasons behind the creation of XHTML:**

**It creates a stricter standard for making web pages, reducing incompatibilities between browsers. So it is compatible for all major browsers.**

**It creates a standard that can be used on a variety of different devices without changes.**

**Let's take an example to understand it.**

**HTML is mainly used to create web pages but we can see that many pages on the internet contain "bad" HTML (not follow the HTML rule).**

**This HTML code works fine in most browsers (even if it does not follow the HTML rules).**

**For example:**

**<html>**

**<head>**

**<title>This is an example of bad HTML</title>**

**<body>**

**<h1>Bad HTML**

**<p>This is a paragraph**

**</body>**

**The above HTML code doesn't follow the HTML rule although it runs. Now a day, there are different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices. The main issue with the bad HTML is that it can't be interpreted by smaller devices.**

**So, XHTML is introduced to combine the strengths of HTML and XML.**

**XHTML is HTML redesigned as XML. It helps you to create better formatted code on your site.**

**XHTML doesn't facilitate you to make badly formed code to be XHTML compatible. Unlike with HTML (where simple errors (like missing out a closing tag) are ignored by the browser), XHTML code must be exactly how it is specified to be.**