

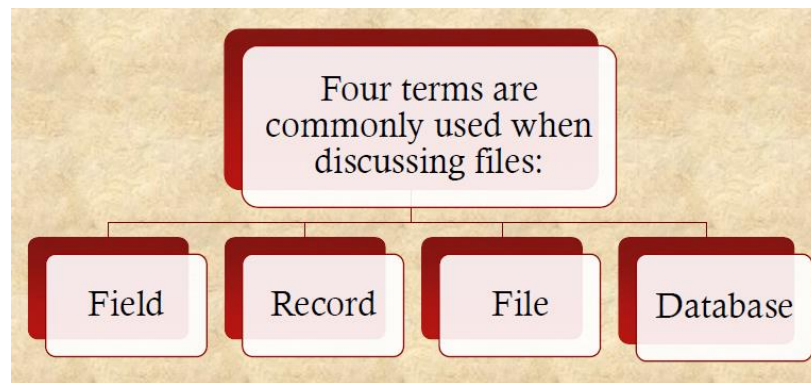
## III B.Sc., [CS] - OS – UNIT V – FILE MANAGEMENT

### FILE – INTRODUCTION

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

### FILE – STRUCTURE

- A File Structure should be according to a required format that the operating system can understand.
- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.



### Structure Terms

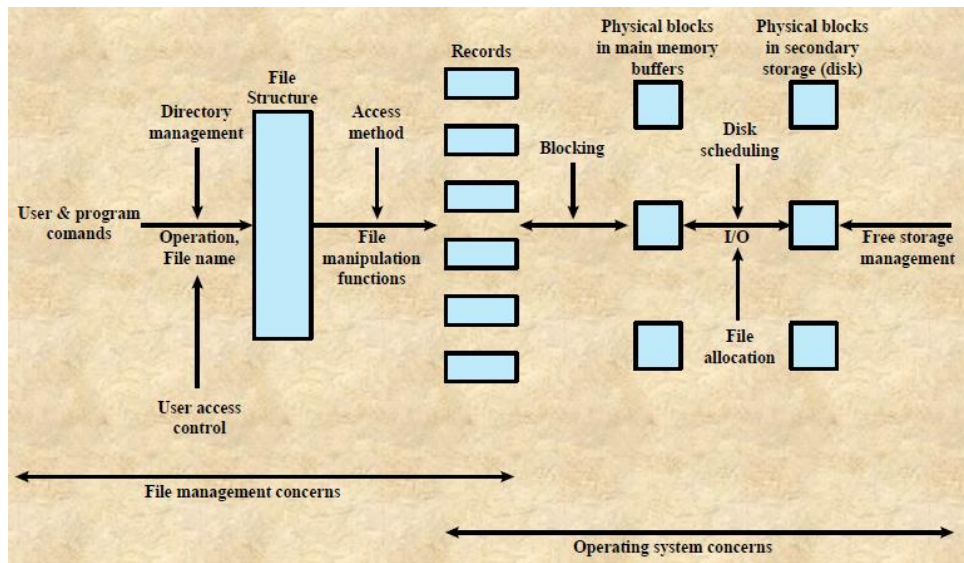
<p><b>Field</b></p> <ul style="list-style-type: none"><li>■ basic element of data</li><li>■ contains a single value</li><li>■ fixed or variable length</li></ul>	<p><b>File</b></p> <ul style="list-style-type: none"><li>■ collection of similar records</li><li>■ treated as a single entity</li><li>■ may be referenced by name</li><li>■ access control restrictions usually apply at the file level</li></ul>
<p><b>Database</b></p> <ul style="list-style-type: none"><li>■ collection of related data</li><li>■ relationships among elements of data are explicit</li><li>■ designed for use by a number of different applications</li><li>■ consists of one or more types of files</li></ul>	<p><b>Record</b></p> <ul style="list-style-type: none"><li>■ collection of related fields that can be treated as a unit by some application program</li><li>■ fixed or variable length</li></ul>

## FILE – TYPES

- File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc.
- Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files
  - Ordinary files
    - These are the files that contain user information.
    - These may have text, databases or executable program.
    - The user can apply various operations on such files like add, modify, delete or even remove the entire file.
  - Directory files
    - These files contain list of file names and other information related to these files.
  - Special files
    - These files are also known as device files.
    - These files represent physical device like disks, terminals, printers, networks, tape drive etc.

## ACCESS METHODS

- Level of the file system closest to the user
- Provides a standard interface between applications and the file systems and devices that hold the data
- Different access methods reflect different file structures and different ways of accessing and processing the data



## FILE ACCESS MECHANISMS

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files –

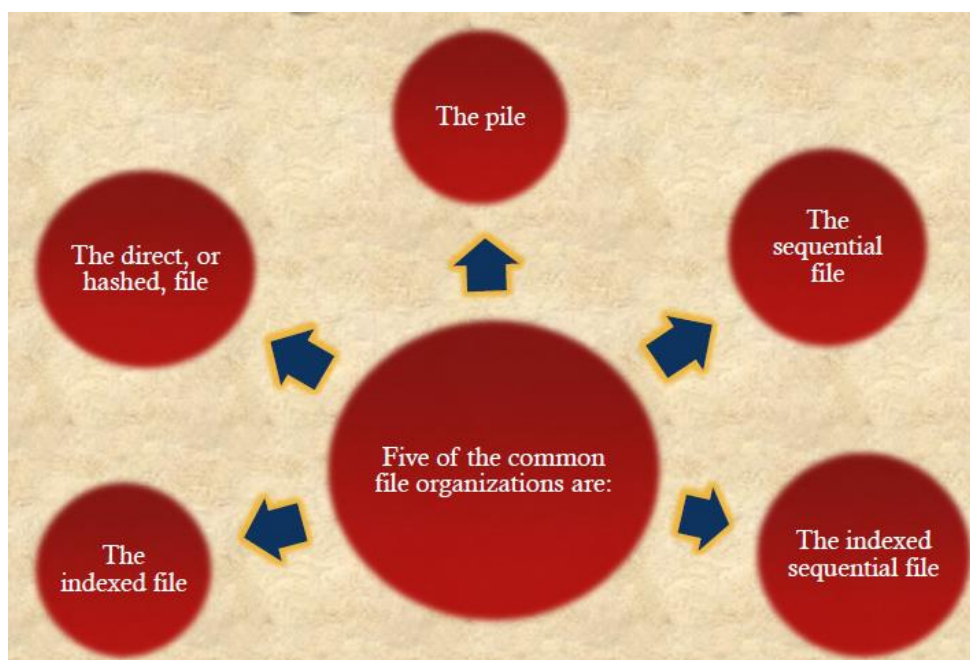
- Sequential access
- Direct/Random access
- Indexed sequential access

- Sequential access
  - A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other.
  - This access method is the most primitive one. Example: Compilers usually access files in this fashion.
- Direct/Random access
  - Random access file organization provides, accessing the records directly.
  - Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
  - The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.
- Indexed sequential access
  - This mechanism is built up on base of sequential access.
  - An index is created for each file which contains pointers to various blocks.
  - Index is searched sequentially and its pointer is used to access the file directly.

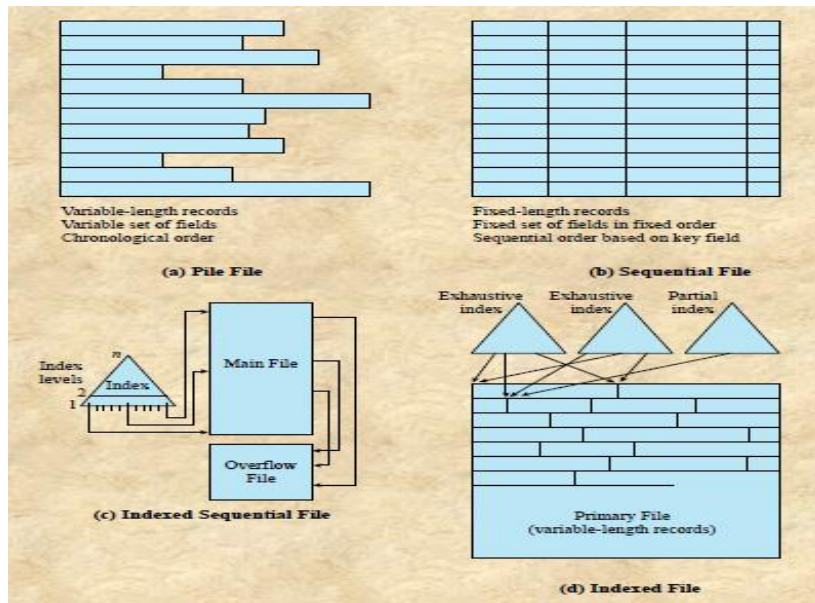
## FILE ORGANIZATION

- File organization is the logical structuring of the records as determined by the way in which they are accessed
- In choosing a file organization, several criteria are important:
  - Short access time
  - Ease of update
  - Economy of storage
  - Simple maintenance
  - Reliability
- Priority of criteria depends on the application that will use the file

## File Organization – Types

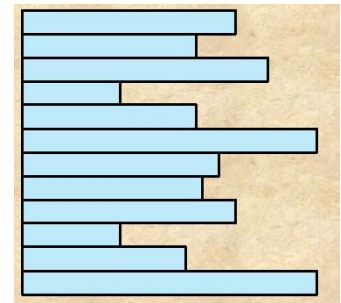


## Common File Organization



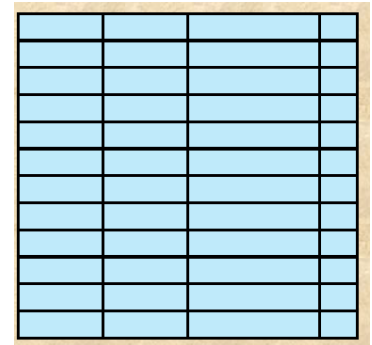
- **The Pile**

- Least complicated form of file organization
- Data are collected in the order they arrive
- Each record consists of one burst of data
- Purpose is simply to accumulate the mass of data and save it
- Record access is by exhaustive search



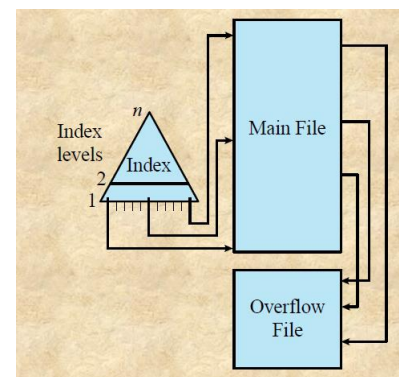
- **The Sequential File**

- Most common form of file structure
- A fixed format is used for records
- Key field uniquely identifies the record
- Typically used in batch applications
- Only organization that is easily stored on tape as well as disk



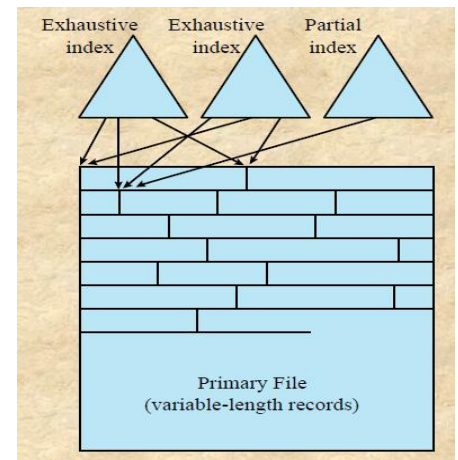
- **Indexed Sequential File**

- Adds an index to the file to support random access
- Adds an overflow file
- Greatly reduces the time required to access a single record
- Multiple levels of indexing can be used to provide greater efficiency in access



- **Indexed File**

- Records are accessed only through their indexes
- Variable-length records can be employed
- Exhaustive index contains one entry for every record in the main file
- Partial index contains entries to records where the field of interest exists
- Used mostly in applications where timeliness of information is critical
- Examples would be airline reservation systems and inventory control systems



## FILE – STORAGE SPACE ALLOCATION

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

- **Contiguous Allocation**

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.
- The directory entry for a file with contiguous allocation contains
  - Address of starting block
  - Length of the allocated portion.

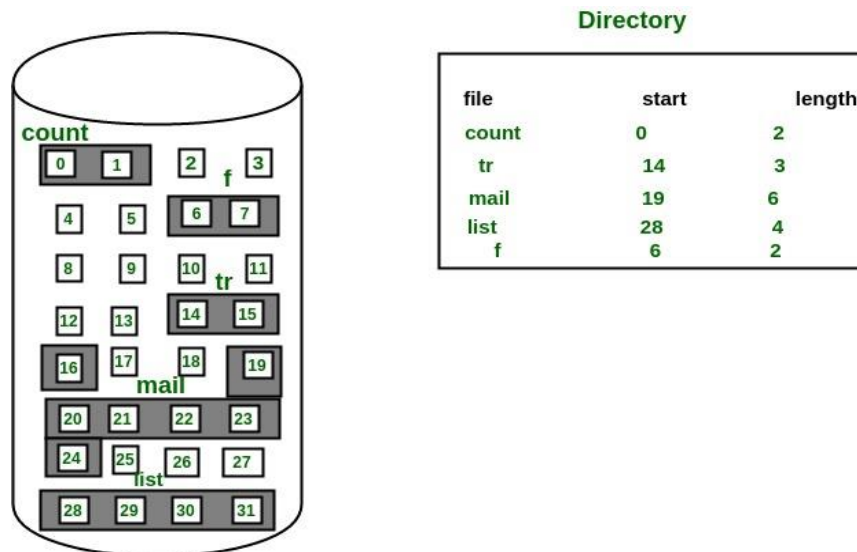
The file 'mail' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.

Advantages:

- Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

Disadvantages:

- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.



**Fig.: Contiguous Allocation**

- **Linked List Allocation**

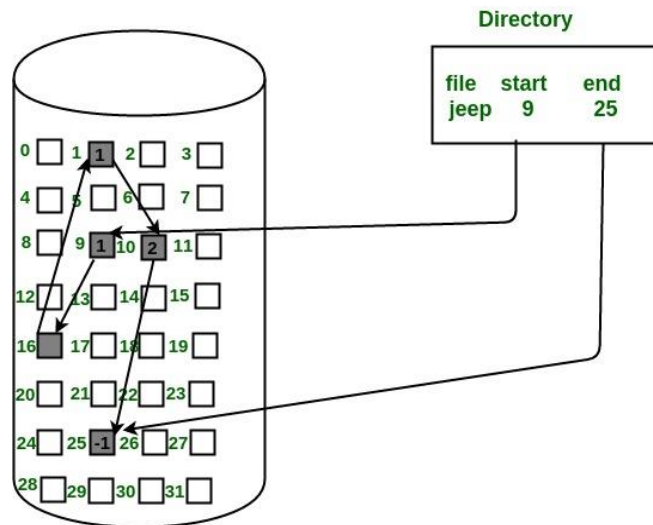
- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.
- In this scheme, each file is a linked list of disk blocks which need not be contiguous.
- The disk blocks can be scattered anywhere on the disk.
- The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

Advantages:

- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

Disadvantages:

- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
- It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access ) from the starting block of the file via block pointers.
- Pointers required in the linked allocation incur some extra overhead.



**Fig.: Linked List Allocation**

- **Indexed Allocation**

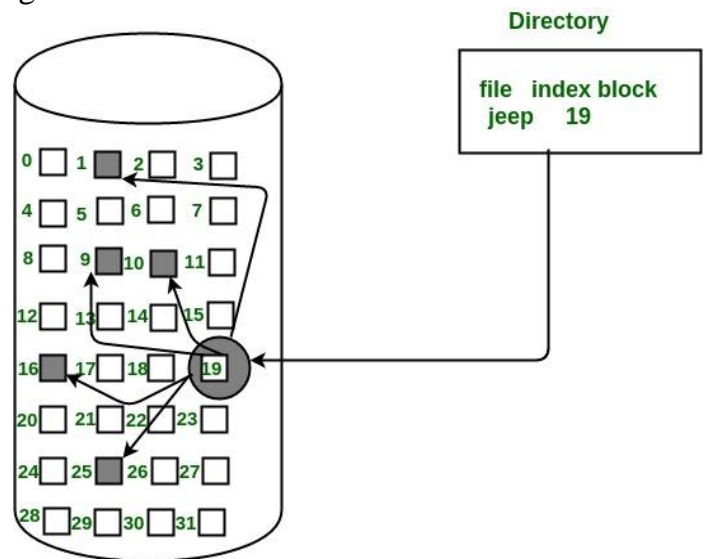
- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.
- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files. In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file.
- Each file has its own index block. The  $i^{\text{th}}$  entry in the index block contains the disk address of the  $i^{\text{th}}$  file block. The directory entry contains the address of the index block as shown in the image:

Advantages:

- This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It overcomes the problem of external fragmentation.

Disadvantages:

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.



**Fig.: Indexed List Allocation**

## III B.Sc., [CS] - PHP – UNIT V – AJAX

### AJAX - INTRODUCTION

AJAX is a developer's dream, because you can:

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

### AJAX Example

HTML Page

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>
</body>
</html>
```

The HTML page contains a <div> section and a <button>.

The <div> section is used to display information from a server.

The <button> calls a function (if it is clicked).

The function requests data from a web server and displays it:

```
function loadDoc()
{
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function()
  {
    if (this.readyState == 4 && this.status == 200)
    {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xmlhttp.open("GET", "ajax_info.txt", true);
  xmlhttp.send();
}
```

### AJAX - THE XMLHttpRequest OBJECT

The keystone of AJAX is the XMLHttpRequest object. All modern browsers support the XMLHttpRequest object. The XMLHttpRequest object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

---

### Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari, Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

```
variable = new XMLHttpRequest();
```

Example

```
var xhttp = new XMLHttpRequest();
```

### Access Across Domains

For security reasons, modern browsers do not allow access across domains. This means that both the web page and the XML file it tries to load, must be located on the same server. If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

---

### Older Browsers (IE5 and IE6)

Old versions of Internet Explorer (5/6) use an ActiveX object instead of the XMLHttpRequest object:

```
variable = new ActiveXObject("Microsoft.XMLHTTP");
```

To handle IE5 and IE6, check if the browser supports the XMLHttpRequest object, or else create an ActiveX object:

Example

```
if (window.XMLHttpRequest) {  
    // code for modern browsers  
    xmlhttp = new XMLHttpRequest();  
}  
else  
{  
    // code for old IE browsers  
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  
}
```

### XMLHttpRequest Object Methods

---

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open( <i>method</i> , <i>url</i> , <i>async</i> , <i>user</i> , <i>psw</i> )	Specifies the request <i>method</i> : the request type GET or POST <i>url</i> : the file location

	<i>async</i> : true (asynchronous) or false (synchronous) <i>user</i> : optional user name <i>psw</i> : optional password
send()	Sends the request to the server Used for GET requests
send( <i>string</i> )	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

---

### XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the <a href="#">Http Messages Reference</a>
statusText	Returns the status-text (e.g. "OK" or "Not Found")

### AJAX - DOWNLOAD IMAGES

AJAX was initially designed to be used with text data (JSON/HTML/XML) and that is why this requirement of downloading images using AJAX were never fulfilled. But with HTML5, XHR2 has been introduced which allows us to get ArrayBuffer in ajax response and this is something which can be used to download image.

There are two approaches that could be taken.

- Download the image, create a blob object and use it as src in img.
- Download the image, and create data url and use it as src in img.

### Download the image, create a blob object and use it as src in img

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function()
{
  if (xhr.readyState===4 && xhr.status===200)
  {
    var blob = new Blob([xhr.response], {type: xhr.getResponseHeader("Content-Type")});
    var imgUrl = window.URL.createObjectURL(blob);
    document.getElementById("img").src = imgUrl;
  }
}
xhr.responseType = "arraybuffer";
xhr.open("GET","Hacker.jpg",true);
xhr.send();
```

### Download the image, and create data url and use it as src in img

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function()
{
  if (xhr.readyState===4 && xhr.status===200)
  {
    document.getElementById("img").src= "data:"+xhr.getResponseHeader("Content-Type")+";
    base64," + btoa(String.fromCharCode.apply(null, new Uint8Array(xhr.response)));
  }
}
xhr.responseType = "arraybuffer";
xhr.open("GET","Hacker.jpg",true);
xhr.send();
```

### AJAX - PHP EXAMPLE

AJAX is used to create more interactive applications. The following example demonstrates how a web page can communicate with a web server while a user types characters in an input field:

#### Example

**Start typing a name in the input field below:**

Suggestions:

First name:

#### Example Explained

In the example above, when a user types a character in the input field, a function called `showHint()` is executed. The function is triggered by the `onkeyup` event.

## HTML code:

```
<html>
<head>
<script>
function showHint(str)
{
  if (str.length == 0)
  {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  else
  {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function()
    {
      if (this.readyState == 4 && this.status == 200)
      {
        document.getElementById("txtHint").innerHTML = this.responseText;
      }
    };
    xmlhttp.open("GET", "gethint.php?q=" + str, true);
    xmlhttp.send();
  }
}
</script>
</head>
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

## Code explanation:

First, check if the input field is empty (`str.length == 0`). If it is, clear the content of the `txtHint` placeholder and exit the function. However, if the input field is not empty, do the following:

- Create an `XMLHttpRequest` object
- Create the function to be executed when the server response is ready
- Send the request off to a PHP file (`gethint.php`) on the server
- Notice that `q` parameter is added `gethint.php?q="+str`
- The `str` variable holds the content of the input field

---

## The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

```
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Honey";
$a[] = "Indu";
$a[] = "John";
$a[] = "King";
$a[] = "Huwa";
$a[] = "Iniya";
$a[] = "Johana";
// get the q parameter from URL
$q = $_REQUEST["q"];
$hint = "";
if ($q !== "")
{
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name)
    {
        if (stristr($q, substr($name, 0, $len))
        {
            if ($hint === "")
            {
                $hint = $name;
            }
            else
            {
                $hint .= ", $name";
            }
        }
    }
}
echo $hint === "" ? "no suggestion" : $hint;
?>
```

## AJAX – DRWING SHAPES

### SVG Shapes

SVG has some predefined shape elements that can be used by developers:

- Rectangle <rect>
- Circle <circle>
- Ellipse <ellipse>
- Line <line>
- Polyline <polyline>
- Polygon <polygon>
- Path <path>

#### (i) RECTANGLE

SVG Rectangle - <rect>

The <rect> element is used to create a rectangle and variations of a rectangle shape:

**Here is the SVG code:**

```
<svg width="400" height="110">
```

```
  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
</svg>
```



#### Code explanation:

The width and height attributes of the <rect> element define the height and the width of the rectangle

- The style attribute is used to define CSS properties for the rectangle
- The CSS fill property defines the fill color of the rectangle
- The CSS stroke-width property defines the width of the border of the rectangle
- The CSS stroke property defines the color of the border of the rectangle

---

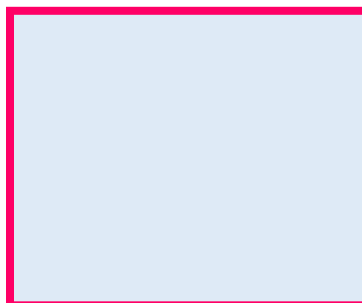
#### Example 2

Let's look at another example that contains some new attributes:

**Here is the SVG code:**

```
<svg width="400" height="180">
```

```
  <rect x="50" y="20" width="150" height="150"
    style="fill:blue;stroke:pink;stroke-width:5;fill-opacity:0.1;stroke-opacity:0.9" />
</svg>
```



## (ii) POLYGON

### SVG Polygon - <polygon>

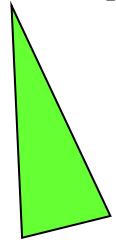
The <polygon> element is used to create a graphic that contains at least three sides. Polygons are made of straight lines, and the shape is "closed" (all the lines connect up). Polygon comes from Greek. "Poly" means "many" and "gon" means "angle".

#### Example 1

The following example creates a polygon with three sides:

#### Here is the SVG code:

```
<svg height="210" width="500">  
  <polygon points="200,10    250,190    160,210" style="fill:lime;stroke:purple;stroke-  
width:1" />  
</svg>
```



#### Code explanation:

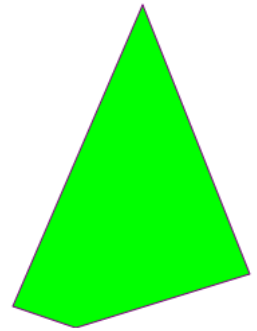
The points attribute defines the x and y coordinates for each corner of the polygon

#### Example 2

The following example creates a polygon with four sides:

#### Here is the SVG code:

```
<svg height="250" width="500">  
  <polygon points="220,10    300,210    170,250  
123,234" style="fill:lime;stroke:purple;stroke-width:1" />  
</svg>
```

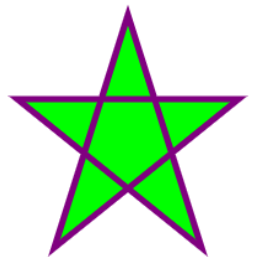


#### Example 3

Use the <polygon> element to create a star:

#### Here is the SVG code:

```
<svg height="210" width="500">  
  <polygon points="100,10    40,198    190,78    10,78    160,198"  
style="fill:lime;stroke:purple;stroke-width:5;fill-rule:nonzero;" />  
</svg>
```



#### Example 4

Change the fill-rule property to "evenodd":

#### Here is the SVG code:

```
<svg height="210" width="500">  
  <polygon points="100,10    40,198    190,78    10,78    160,198"  
style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />  
</svg>
```

