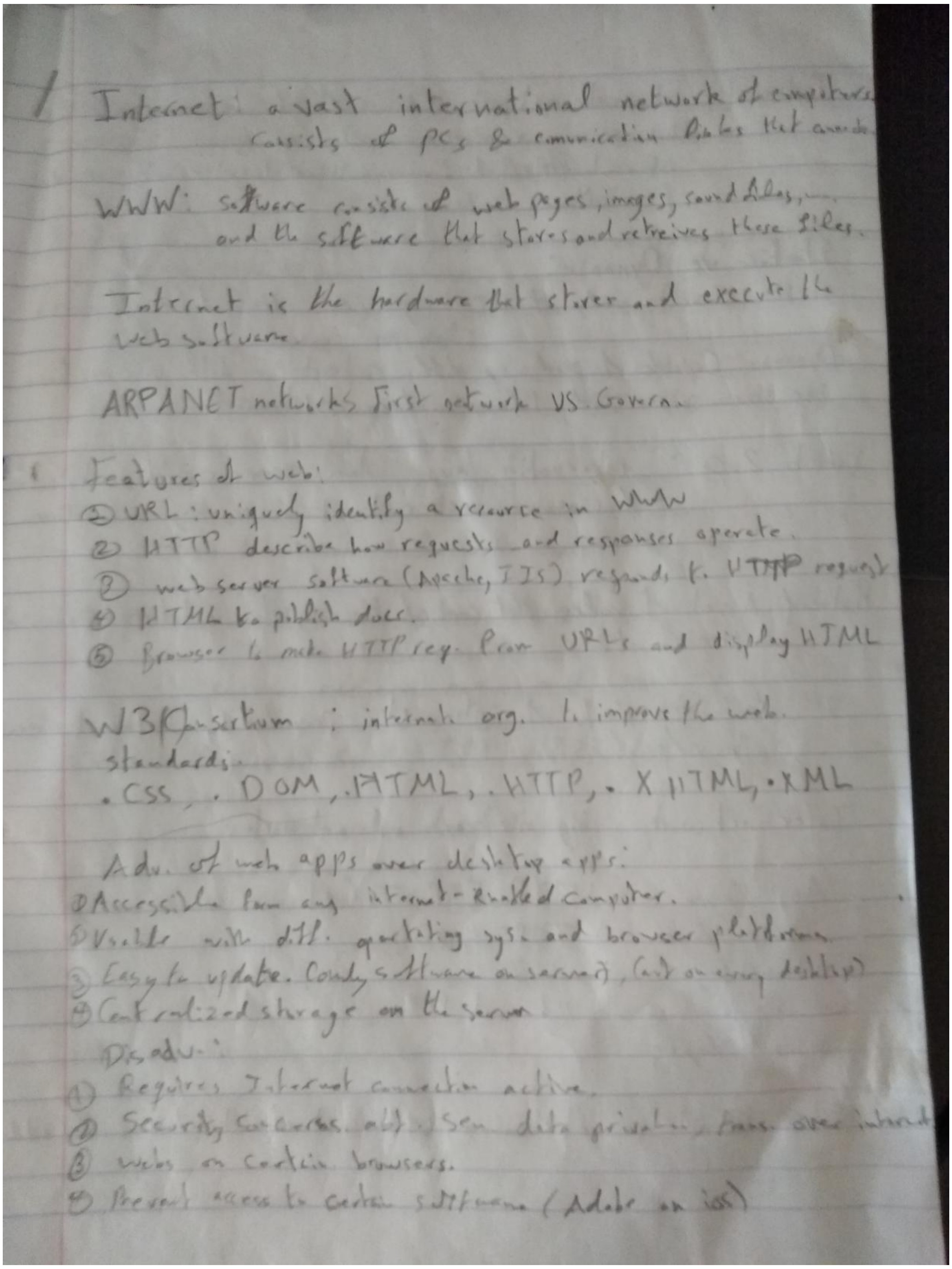


Internet: a vast international network of computers
consists of PCs & communication links that connects

WWW: software consists of web pages, images, sound files, ...
and the software that stores and retreives these files.

Internet is the hardware that stores and execute the web software.

ARPANET networks first network VS Govern.

Features of web:
① URL: uniquely identify a resource in WWW
② HTTP describe how requests and responses operate.
③ web server software (Apache, IIS) respond to HTTP request
④ HTML to publish docs.
⑤ Browser to make HTTP req. from URLs and display HTML

W3Consortium: internat. org. to improve the web.
standards:
. CSS, . DOM, . HTML, . HTTP, . XHTML, . XML

Adv. of web apps over desktop apps:
① Accessible from any internet-enabled computer.
② Usable with diff. operating sys. and browser platforms.
③ Easy to update. (only software on server), (not on every desktop)
④ Centralized storage on the server

Disadv.:
① Requires Internet connection active.
② Security concerns abt. sen. data private, trans. over internet
③ works on certain browsers.
④ Prevent access to certain software (Adobe on ios)

Intranet: internal network that is local to an org or bus has private resources (only employees have access) internal protected from unauthorized external users by firewalls.

Static vs Dynamic:
static: Consists only of HTML pages (that identical for all users)

Dynamic: Capable of producing diff. content for diff. visitors from the same source code file (location, OS/browser)

Web 2.0: interactive exp. where users could contribute and consume web content. creating user-driven web exp.

Protocol: A set of rules that control the way two entities communicate with each other.

IP: identifies dest. of internet. provided by Internet Service Provider (ISP)

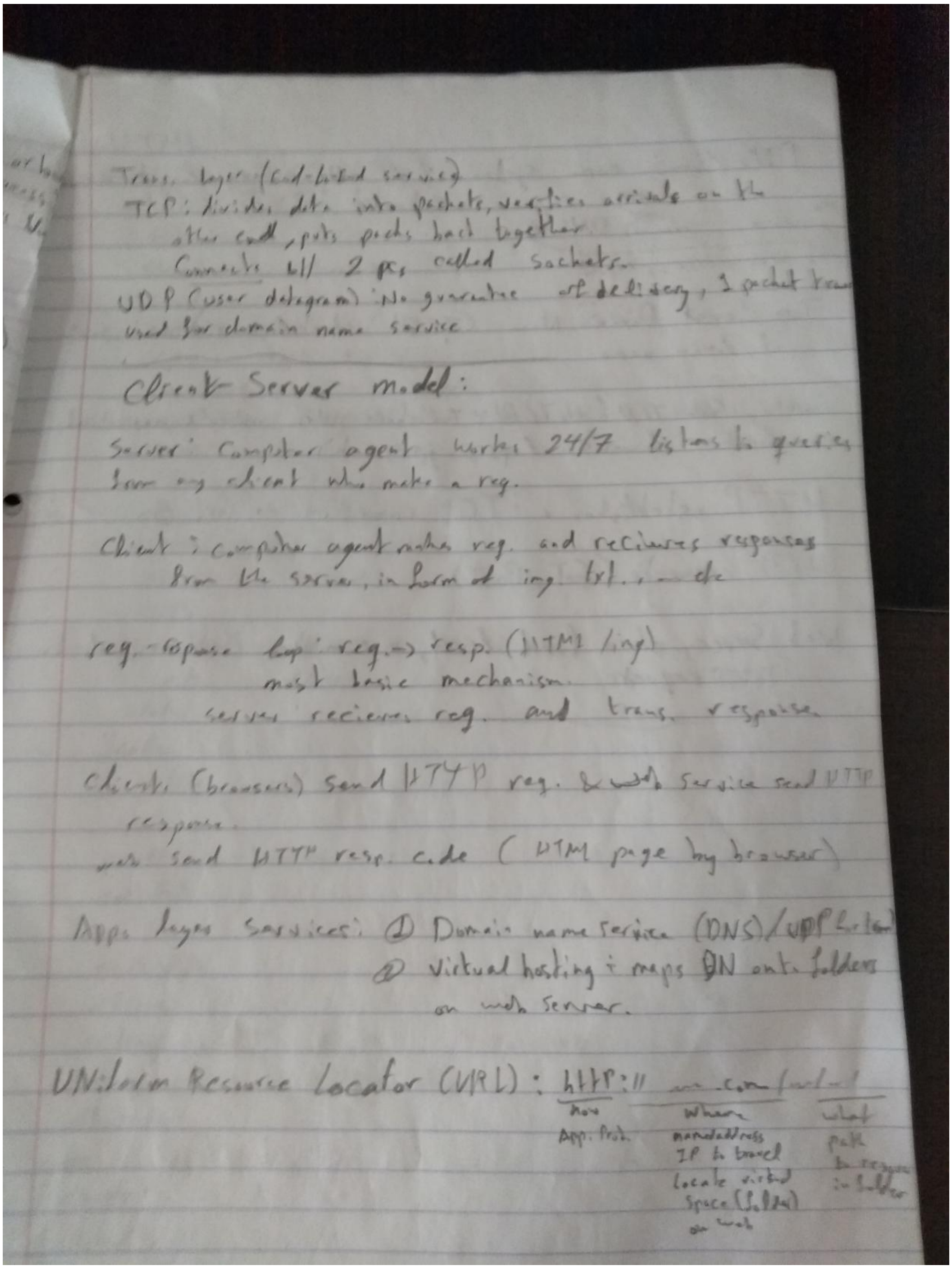
Inter-network: big network of networks

App layer → HTTP, FTP, Telnet
Transp. layer → TCP, UDP
InterNetwork layer → $\overset{\text{Trans ctrl}}{\text{IP}}$
Network interface → Ethernet, PPP
Physical layer

Routers direct packets among diff networks based upon IP address.

Trans.
TCP:

UD



Trans. layer (end-to-end service)
TCP: divides data into packets, verifies arrivals on the
other end, puts packets back together
Connects b/1 2 pcs called sockets.
UDP (user datagram): No guarantee of delivery, 1 packet trans
used for domain name service

Client Server model:

Server: Computer agent works 24/7 listens to queries
from any client who make a req.

Client: computer agent makes req. and recieves responses
from the server, in form of img, txt, ... etc

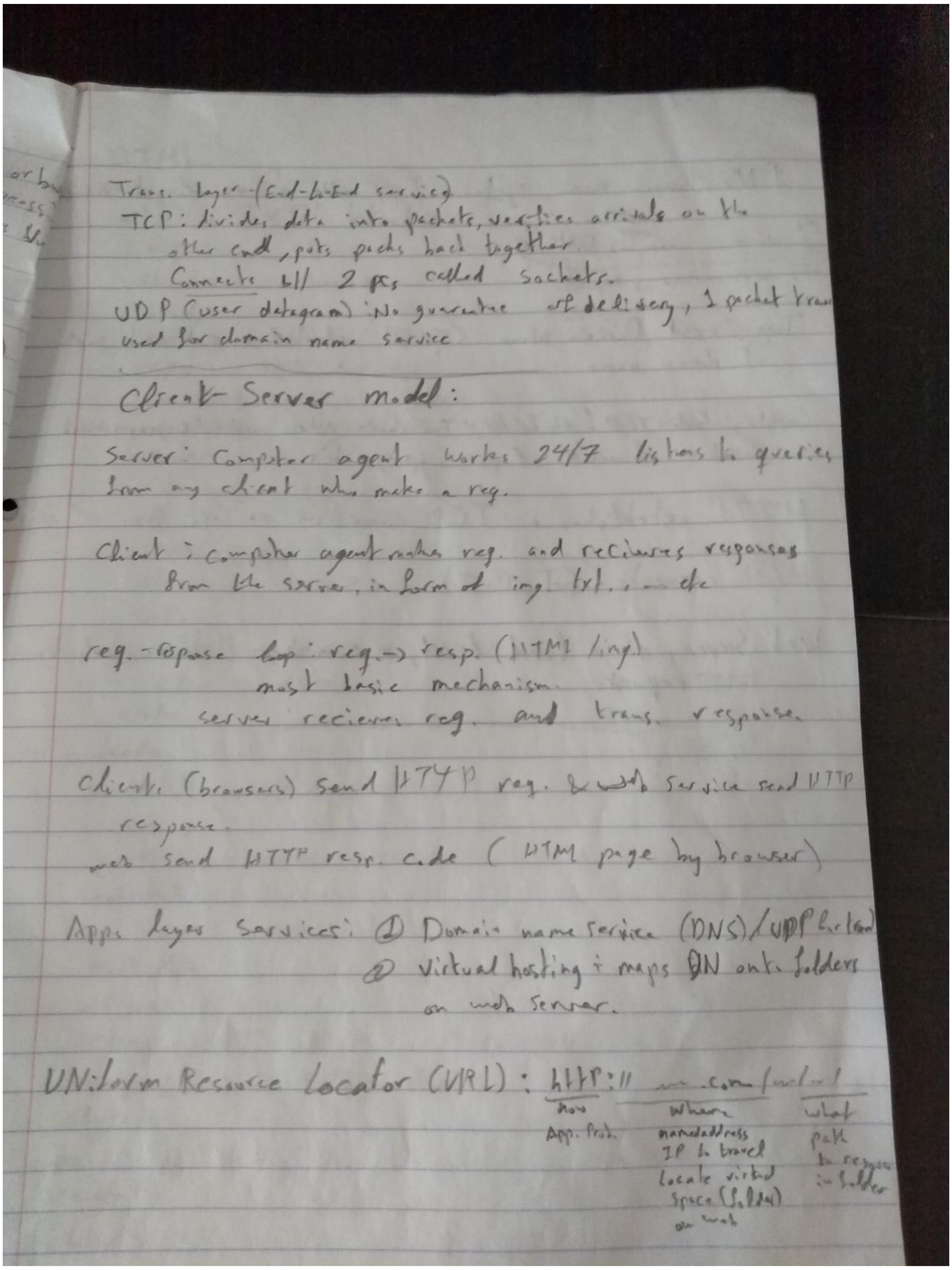
req.-response loop: req. → resp. (HTML / img)
most basic mechanism.
server recieves req. and trans. response.

Clients (browsers) send HTTP req. & web service send HTTP
response.
web send HTTP resp. code (HTML page by browser)

Apps layer services: ① Domain name service (DNS) / UDP [illegible]
② Virtual hosting: maps DN onto folders
on web server.

UNiform Resource Locator (URL): $\underbrace{\text{http://}}_{\text{how}}$ $\underbrace{\text{...com}}_{\text{where}}$ $\underbrace{\text{/.../.../}}_{\text{what}}$

how: App. Prot.
where: name/address, IP to travel, locate virtual space (folder) on web
what: path to resource in folder



Trans. layer (End-to-End service)
TCP: divides data into packets, verifies arrivals on the
other end, puts packs back together
Connects b// 2 pcs called sockets.
UDP (user datagram): No guarantee of delivery, 1 packet tran
used for domain name service

Client Server model:

Server: Computer agent works 24/7 listens to queries
from any client who make a req.

Client: computer agent makes req. and recieves responses
from the server, in form of img. txt., ... etc

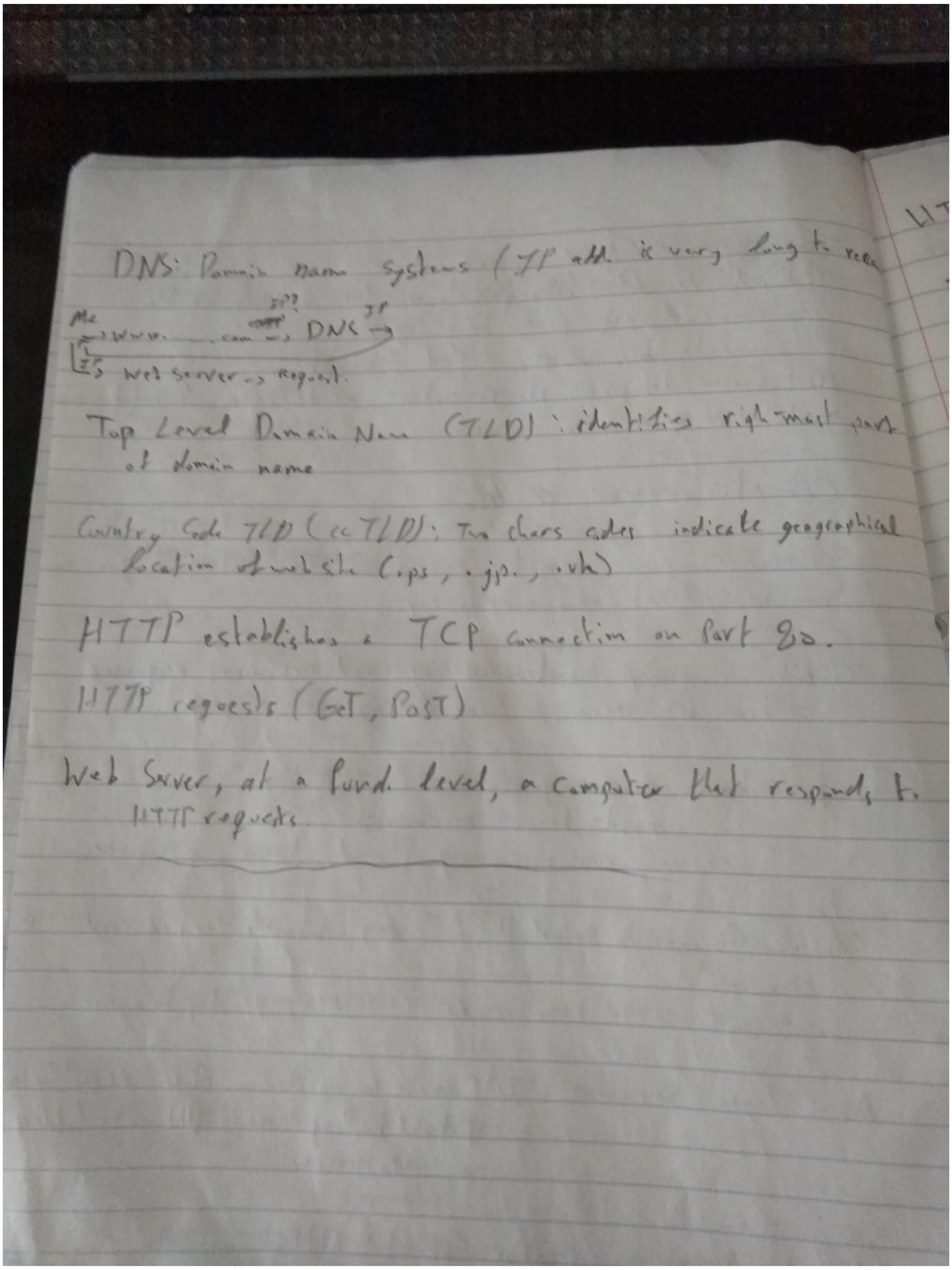
req.-response loop: req. -> resp. (HTML / img)
most basic mechanism.
server recieves req. and trans. response.

clients (browsers) send HTTP req. & web service send HTTP
response.
web send HTTP resp. code (HTML page by browser)

Apps layer services: ① Domain name service (DNS) / UDP [illegible]
② virtual hosting : maps DN onto folders
on web server.

UNiform Resource Locator (URL) : HTTP:// ....com/..../..../

HTTP://	....com	/..../..../
how	Where	what
App. Prot.	name/address IP to travel locate virtual Space (folder) on web	path to resource in folder



DNS: Domain name systems (IP add. is very long to rem

Me → www.____.com $\xrightarrow{IP?}$ DNS $\xrightarrow{IP}$

↳ IP → web server → request.

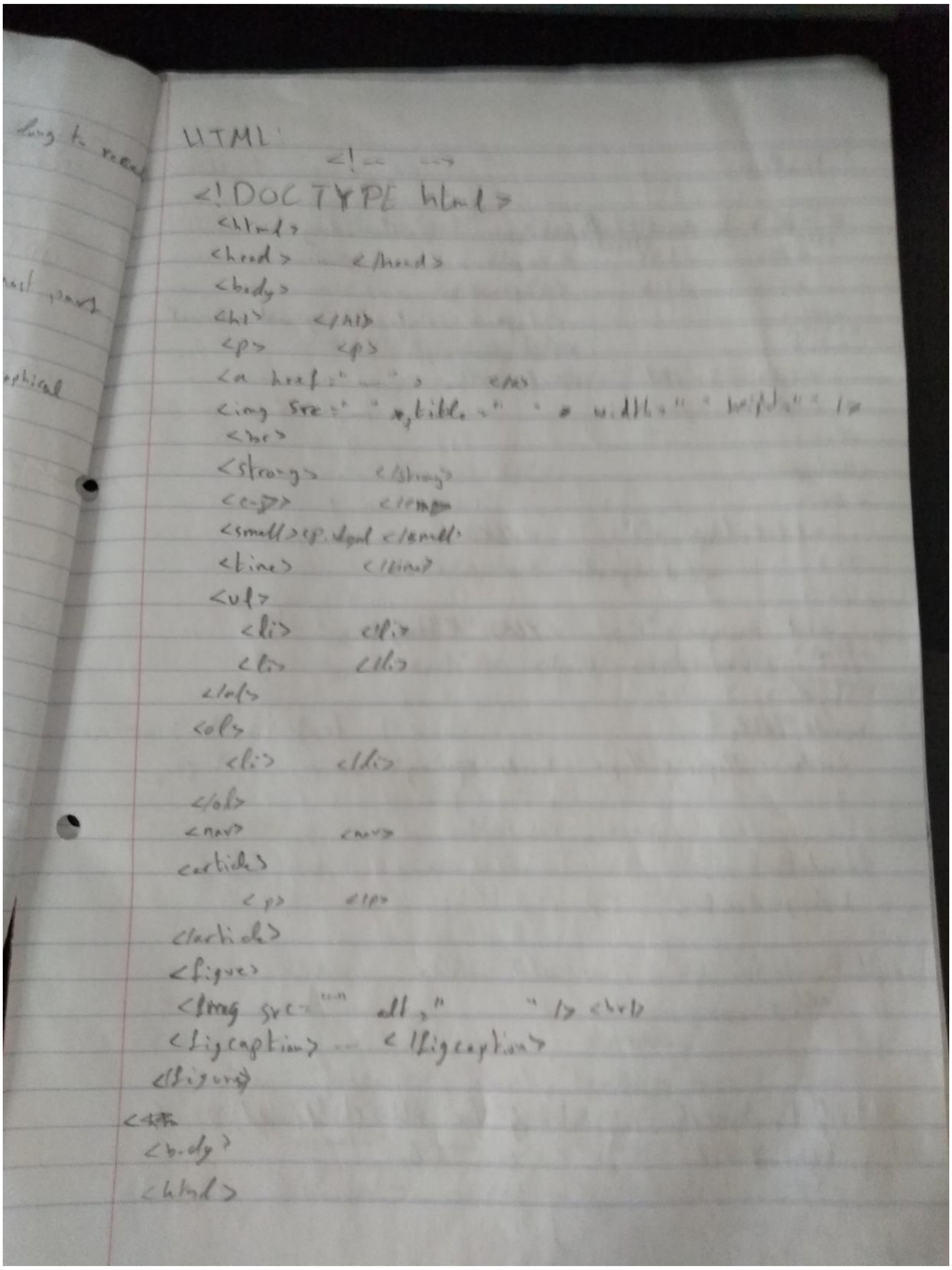
Top Level Domain Name (TLD): identifies right-most part of domain name

Country Code TLD (ccTLD): Two chars codes indicate geographical location of website (.ps, .jp, .uk)

HTTP establishes a TCP connection on Port 80.

HTTP requests (GET, POST)

Web Server, at a fund. level, a computer that responds to HTTP requests.



long to recall

nal part

phical

HTML:

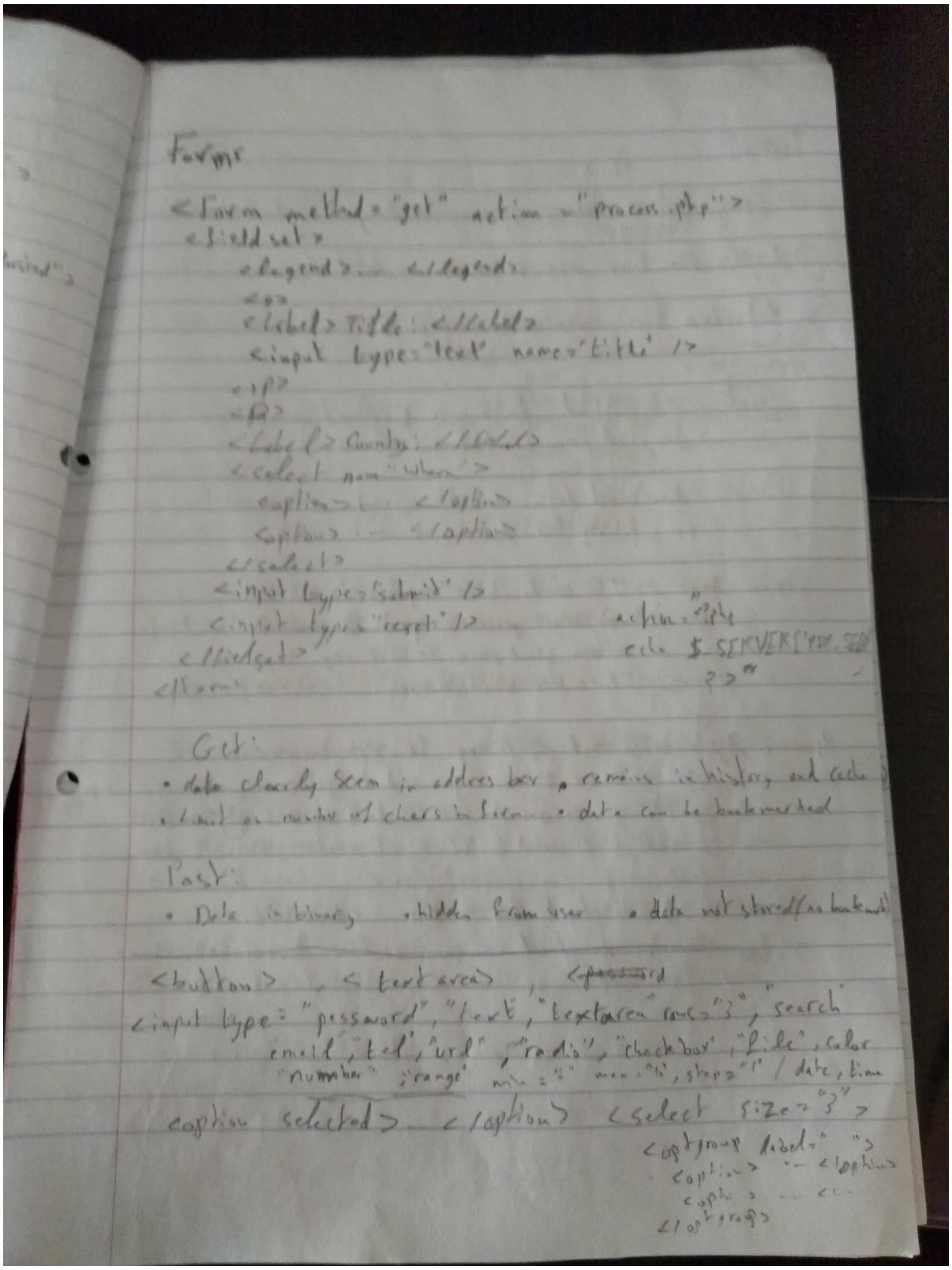
```
            <!--   -->
<!DOCTYPE html>
  <html>
  <head>    </head>
  <body>
  <h1>   </h1>
   <p>     <p>
  <a href="..." >      </a>
  <img src=" " title=" " width=" " height=" " />
   <br>
  <strong>     </strong>
  <em>       </em>
  <small>cp. dyd </small>
  <time>      </time>
  <ul>
    <li>      </li>
    <li>      </li>
  </ul>
 <ol>
   <li>      </li>
 </ol>
 <nav>          <nav>
 <article>
      <p>     </p>
 </article>
 <figure>
 <img src="" alt="           " /> <hr/>
 <figcaption> ...  </figcaption>
<figure>
<[illegible]
 <body>
 <html>
```

Tables

```
<Table border=1 cellspacing=" " cell pading=" ">
 <tr>                 <thead>
   <th>    <th>          | <caption> ... </caption>
   <th> .. </th>         | <colgroup>
                         | <col span="2" style="background-color:red">
 </tr>                   | <col ~~span~~ style=" ">
<tbody>                  | </colgroup>
 <tr>
   <td> ... </td>
   <td> ... </td>
 </tr>
~~</table>~~
<tr>
   <td colspan="2">   </td>
</tr>        <td bgcolor=" "> .. </td>
<tr>
~~<tr>~~ <td rowspan="2"> .. </td>  </tbody>
 <tfoot>
 <tr> <td></td> </tr> </tfoot>
<Style>
~~borde~~ table {                        text-transform: uppercase;
  border-collapse: collapse; border-spacing: horz.px  vert.px;
}

thead tr {
  background-color: " ";

th {
  padding: n em;
}

tbody tr:hover {          tbody tr:nth-child(odd)/(2) {
  background-color: ..       background-color:   ;
  color:   ;              }
}
```

Forms

<For
<



Forms

```
<form method="get" action="process.php">
 <fieldset>
    <legend>... </legend>
    <p>
    <label>Title: </label>
    <input type="text" name="title" />
   </p>
   <p>
   <label> Country: </label>
   <select name="where">
     <option> ... </option>
     <option> ... </option>
   </select>
  <input type="submit" />
  <input type="reset" />
 </fieldset>
</form>
```

```
action="<?php
echo $_SERVER['PHP_SEL
?>"
```

Get:

- data clearly seen in address bar
- remains in history and cache
- limit on number of chars in form
- data can be bookmarked

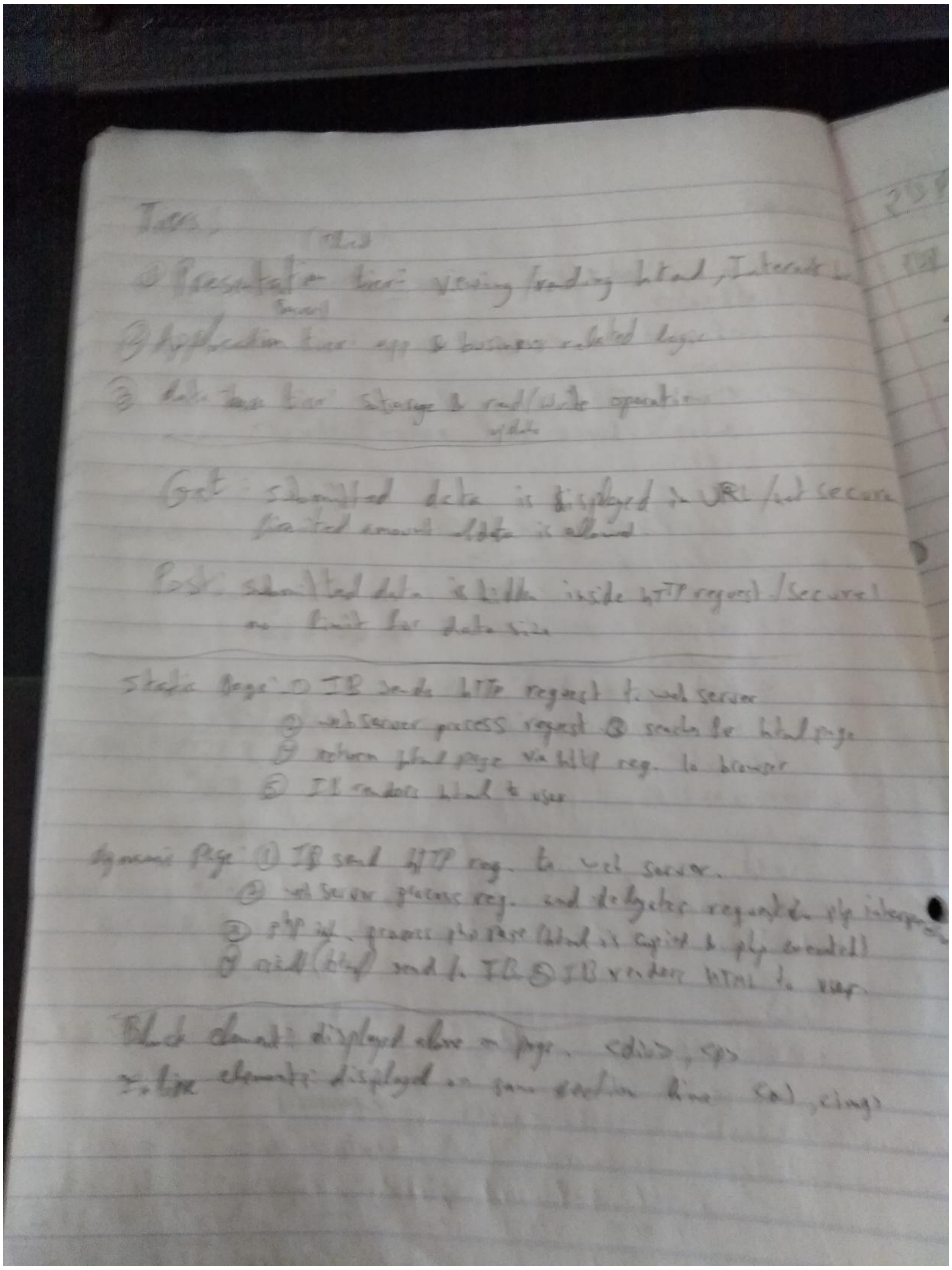
Post:

- Data in library
- hidden from user
- data not stored (no bookmark)

<button> , < text area> , <~~password~~

<input type="password", "text", "textarea" rows="3", "search"
email, tel, "url", "radio", "checkbox", "file", color
"number" "range" min="1" max="10", step="1" / date, time

<option selected> ... </option> <select size="3">
<optgroup label=" ">
<option> ... </option>
<option> ... </...
</optgroup>



Tiers,

(html)
① Presentation tier: viewing/reading html, Interacts [illegible]
(server)
② Application tier: app & business related logic
③ data base tier: storage & read/write operations
w/data

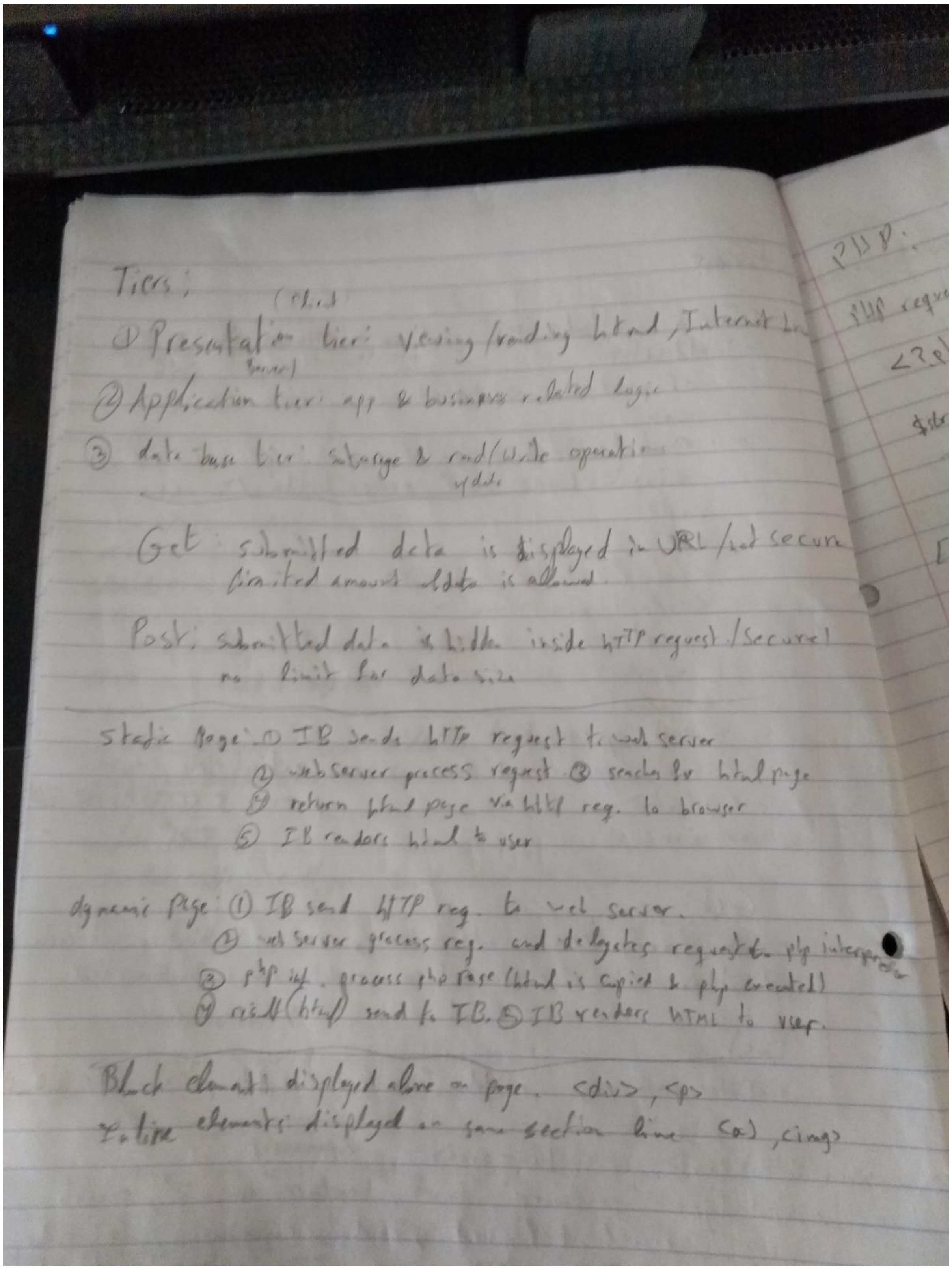
Get: submitted data is displayed in URL/not secure
limited amount of data is allowed

Post: submitted data is hidden inside HTTP request/secure
no limit for data size

Static Page: ① IB sends HTTP request to web server
② webserver process request ③ searches for html page
④ return html page via HTTP req. to browser
⑤ IB renders html to user

Dynamic Page: ① IB send HTTP req. to web server.
② web server process req. and delegates request to php interpreter
③ php int. process php page (html is copied & php executed)
④ result (html) send to IB ⑤ IB renders HTML to user.

Block elements: displayed alone on page. <div>, <p>
Inline elements: displayed on same section line <a>, <img>



Tiers;

① Presentation tier (client): viewing/reading html, Internet br

② Application tier (server): app & business related logic

③ data base tier: storage & read/write operations update

Get: submitted data is displayed in URL / not secure
limited amount of data is allowed.

Post: submitted data is hidden inside HTTP request / secure
no limit for data size

static page: ① IB sends HTTP request to web server
② web server process request ③ searches for html page
④ return html page via HTTP req. to browser
⑤ IB renders html to user

dynamic page: ① IB send HTTP req. to web server.
② web server process req. and delegates request to php interpreter
③ php int. process php page (html is copied & php executed)
④ result (html) send to IB. ⑤ IB renders HTML to user.

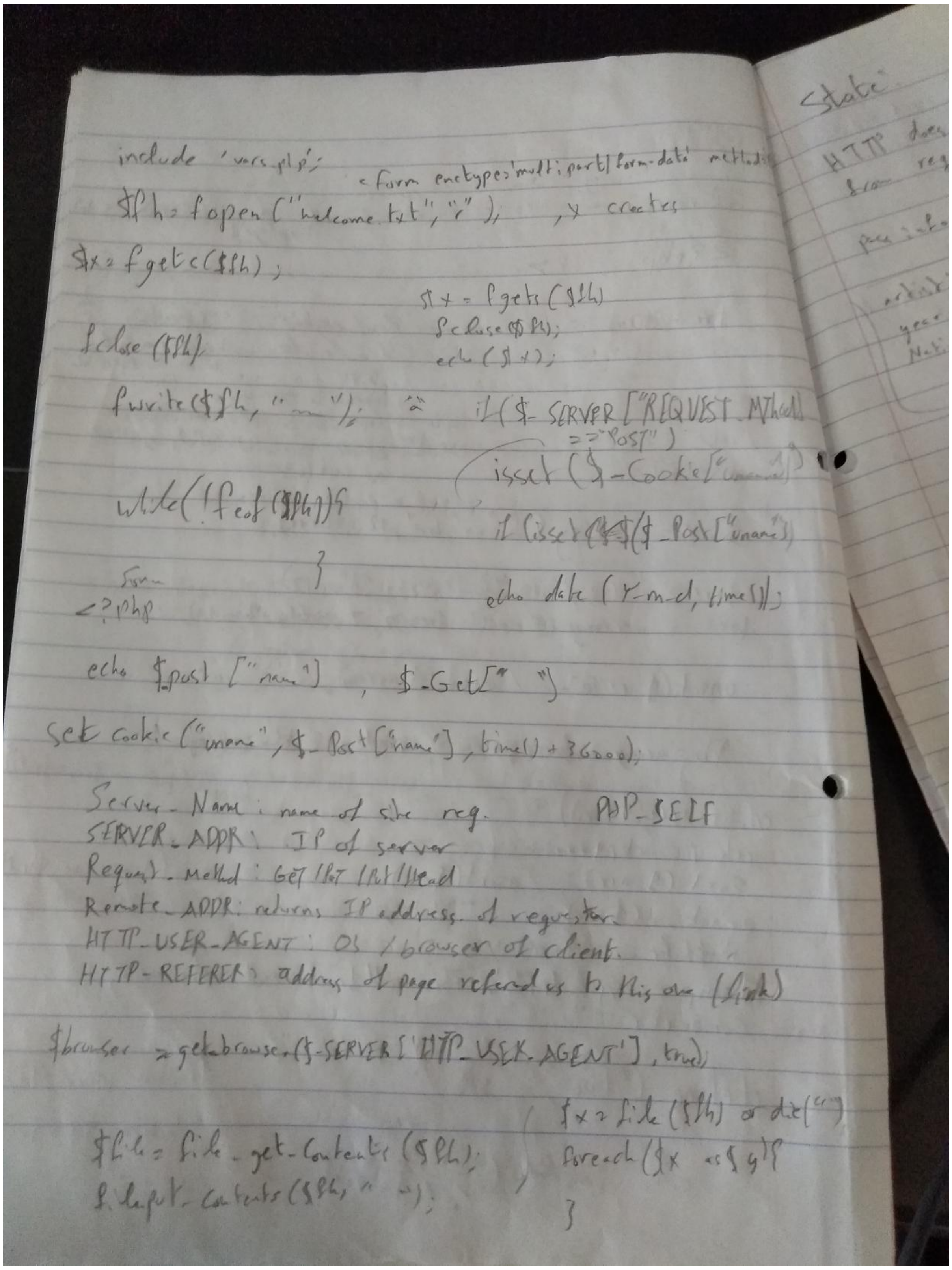
Block elements: displayed alone on page. <div>, <p>
Inline elements: displayed on same section line <a>, <img>

PHP:
php reque
<?p
$str

PHP:

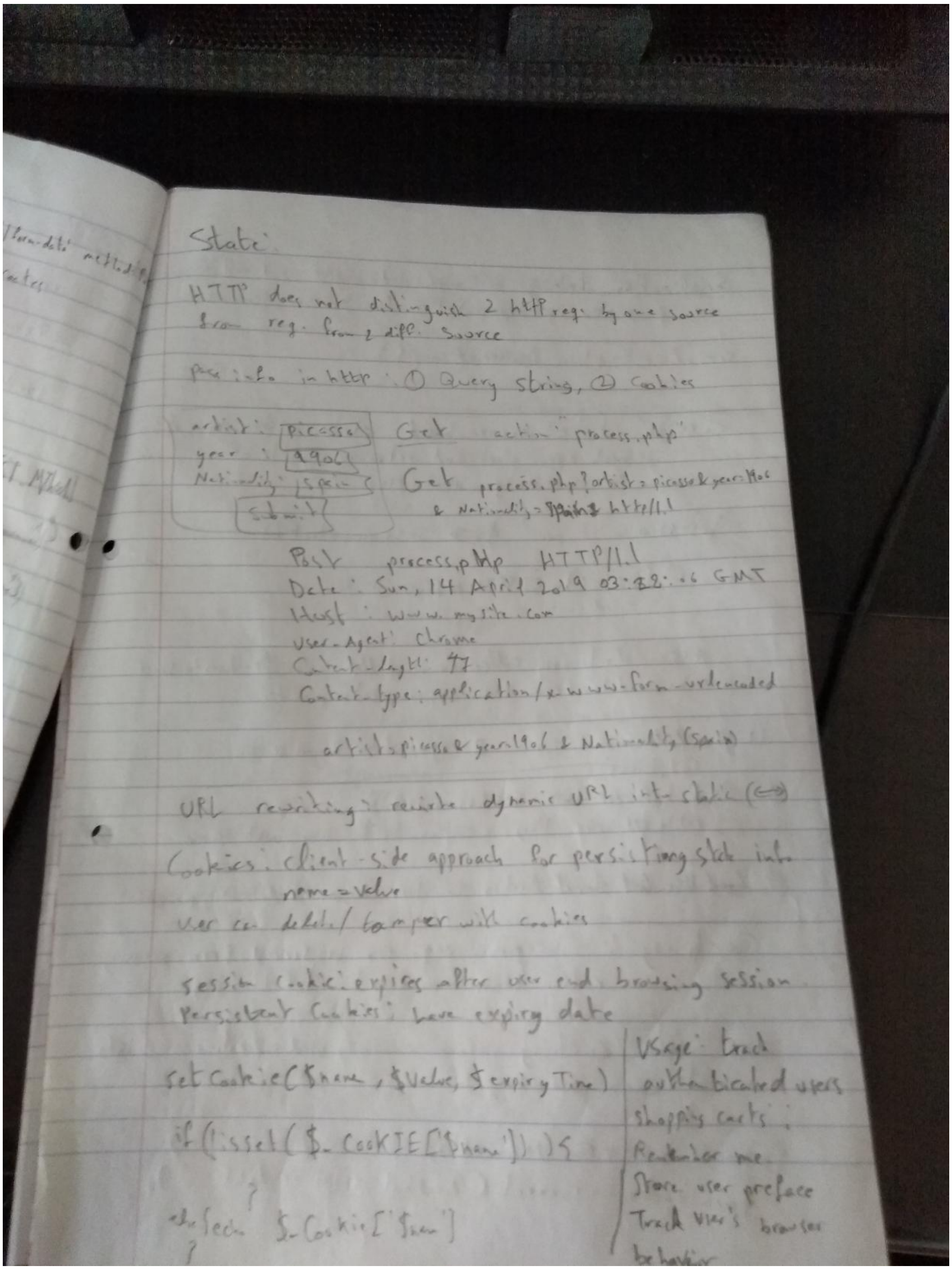
PHP request $\xrightarrow{\text{auto}}$ web server $\xrightarrow{\text{execute}}$ Browser displayed (server side)

```
<?php       ?>

$str = EOD                         Zend engine:        Lexer
  ~~~~                              PHP code str. -> human-readable
  ~~~~                                               to machine-digestible
  ~~~~                              Parser                 token
                                    -> tokens to expressions
EOD;                     Compile -> expressions to PHP opcode (bytecode)
echo $str;               executes -> opcodes, executed safely -> VM
                                    -> sent back to browser

$arr = array("foo" => "bar", 12 => true);

$arr = array(5 => 43, 32, 56, "b" => 12);
$arr = array(5 => 43, 6 => 32, 7 => 56, "b" => 12);

unset($arr[e]); / unset($arr[..]);

$b = array_values($a)   // reindexes the array

echo (is_array($arr)) ? "..." : "..." ;         foreach($arr as $key => $val)
 echo $count($arr);                                {
 sort($arr); / asort($arr)                         }
                    assoc. keys                   foreach($arr as $val)
 print_r($arr);                                   { echo "$value<br>";
 shuffle($arr)                                    }
$temp = explode(' ', "This will be turned into array");
$temp = compact(" ", " ", " ")

define("foo", "sk");                              function name($x){
                                                      return $x * $x
   echo implode(',', $_POST[...])                 }
```



```
include 'vars.php';
                    <form enctype='multipart/form-data' method
$fh = fopen("welcome.txt", "r");        , x creates
$x = fgetc($fh);
                              $x = fgets($fh)
fclose($fh)                   fclose($fh);
                              echo($x);
fwrite($fh, "   ");      "a"      if($_SERVER["REQUEST_Method"]
                                        =="POST")
                                   isset($_Cookie["uname"])
while(!feof($fh)){
                                   if(isset(~~($~~($_Post["uname"])
form             }
<?php                              echo date(Y-m-d, time());

echo $_post["name"]     , $_Get["  "]

set cookie("uname", $_Post["name"], time()+36000);

Server_Name : name of site req.          PHP_SELF
SERVER_ADDR : IP of server
Request_Method : GET/Post/Put/Head
Remote_ADDR : returns IP address of requester.
HTTP_USER_AGENT : OS / browser of client.
HTTP_REFERER : address of page refered us to this one (link)

$browser = get_browser($_SERVER['HTTP_USER_AGENT'], true);

                                         $x = file($fh) or die("")
$file = file_get_Contents($fh);          foreach($x as $y){
file_put_Contents($fh, "  ");
                                         }
```

State:
HTTP does
from req



/form-data' method

racters

T Method

State:

HTTP does not distinguish 2 http req. by one source from req. from 2 diff. source

pass info. in http: ① Query string, ② Cookies

artist: Picasso
year: 1906
Nationality: Spain
Submit

Get action: "process.php"

Get process.php?artist=picasso&year=1906 & Nationality=~~Spain~~ http/1.1

```
Post  process.php  HTTP/1.1
Date: Sun, 14 April 2019 03:32:06 GMT
Host: www.mysite.com
User-Agent: chrome
Content-length: 47
Content-type: application/x-www-form-urlencoded

    artist=picasso&year=1906 & Nationality(Spain)
```

URL rewriting: rewrite dynamic URL into static (⟺)

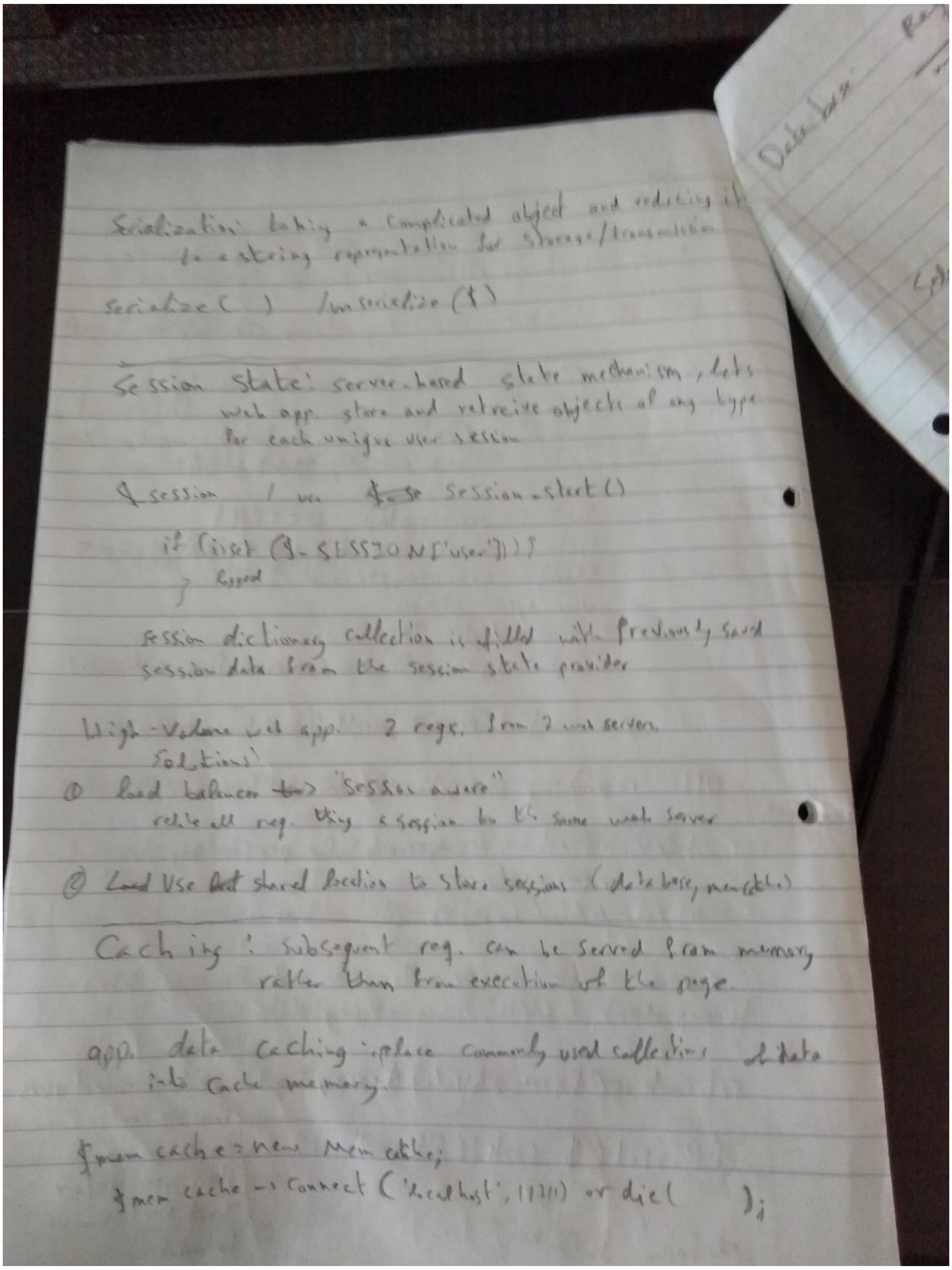
Cookies: client-side approach for persisting state info.
name = value

user can delete/tamper with cookies

session cookie: expires after user end browsing session
Persistent Cookies: have expiring date

```
setCookie($name, $value, $expiryTime)

if (isset($_COOKIE['$name'])) {

    }
else {  $_Cookie['$name']
}
```

Usage: track authenticated users
shopping carts;
Remember me.
Store user preface
Track user's browser behavior



Serialization: taking a complicated object and reducing it to a string representation for storage/transmission

serialize( ) /unserialize ($)

Session state: server-based state mechanism, lets web app. store and retreive objects of any type for each unique user session

$session / use ~~$_se~~ session_start()

```
if (isset ($_SESSION['user'])) {
} logged
```

session dictionary collection is filled with previously saved session data from the session state provider

High-Volume web app. 2 reqs. from 2 web servers.
Solutions:

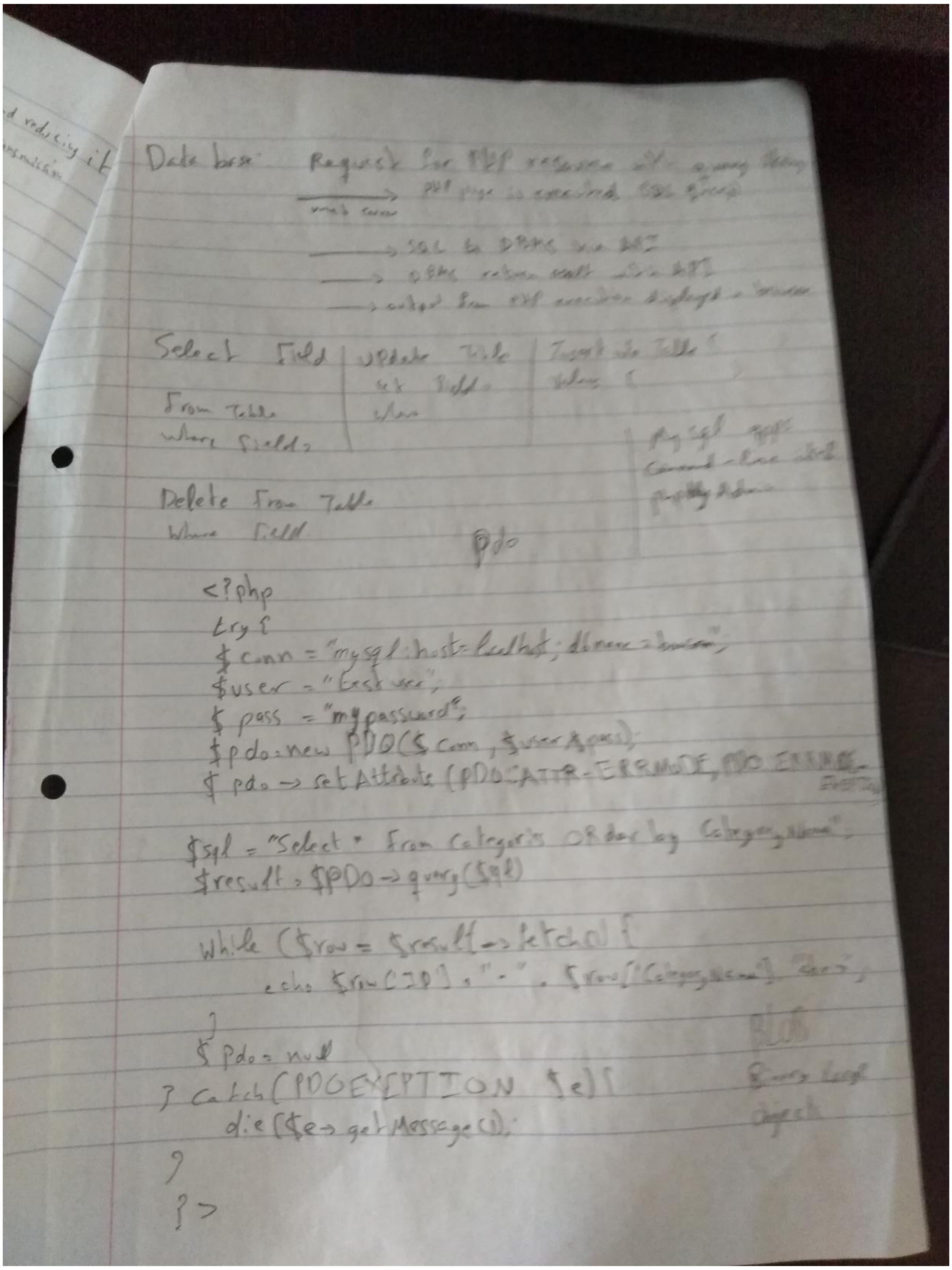
① load balancer ~~to~~> "session aware"
route all req. using a session to the same web server

② ~~Load~~ Use ~~that~~ shared location to store sessions (database, memcache)

Caching: subsequent req. can be served from memory rather than from execution of the page.

app. data caching: place commonly used collections of data into cache memory.

```
$memcache = new Memcache;
  $memcache -> connect ('localhost', 11211) or die(      );
```



d reducing it
transmission

Data base: Request for PHP resource [illegible] query string
$\xrightarrow[\text{web server}]{}$ PHP page is executed SQL [illegible]
⟶ SQL to DBMS via API
⟶ DBMS returns result via API
⟶ output from PHP execution displayed in browser

Select Field From Table where Field =	UPdate Table set Field = where	Insert into Table (Values ( )

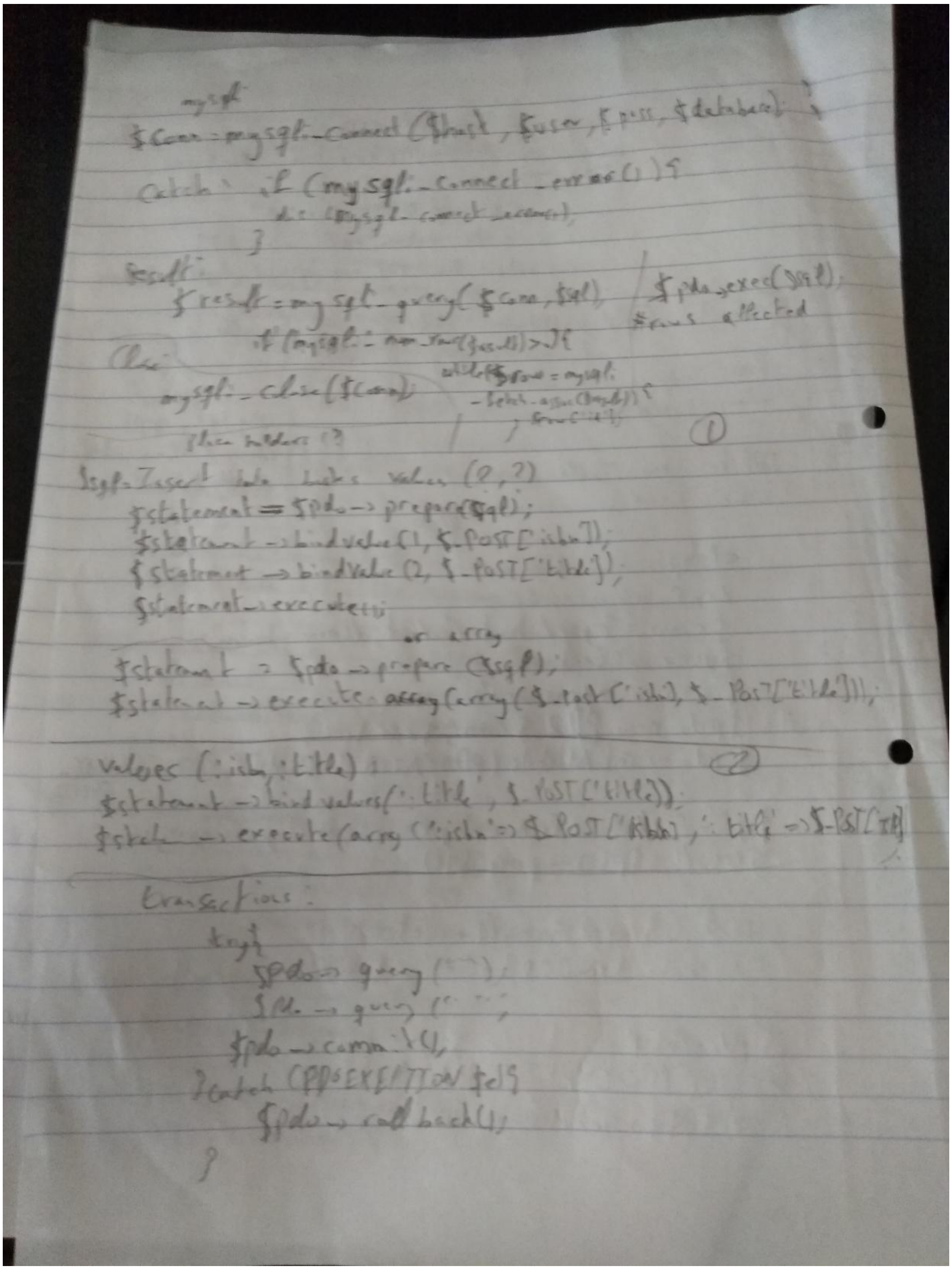
Delete From Table
Where Field

MySql apps
Command-line client
phpMyAdmin

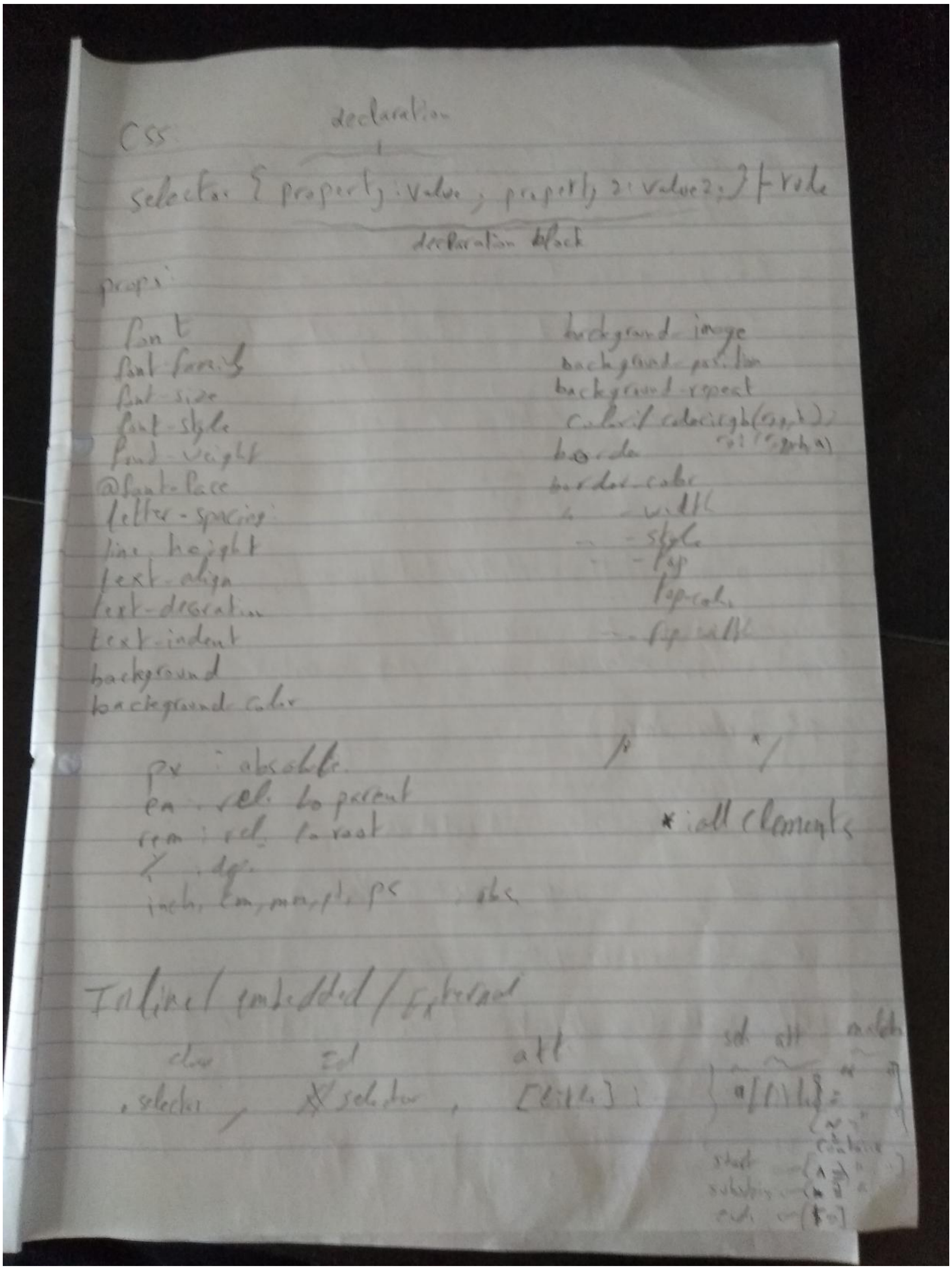
pdo

```
<?php
try {
$conn = "mysql:host=localhost; dbname=[illegible]";
$user = "test user";
$pass = "mypassword";
$pdo = new PDO($conn, $user, $pass);
$pdo -> setAttribute(PDO::ATTR-ERRMODE, PDO::ERRMODE_EXCEPTION

$sql = "Select * From Categories ORder by Category_Name";
$result = $PDO -> query($sql)

While ($row = $result -> fetch()) {
    echo $row['ID'] . " - " . $row['Category_Name'] . "<br>";
}
$pdo = null
} Catch(PDOEXCEPTION $e) {
    die($e -> getMessage());
}
?>
```

BLOB
Binary Large
object



```
mysqli
$conn = mysqli_connect($host, $user, $pass, $database);

Catch:  if (mysqli_connect_errno()){
            die (mysqli_connect_error());
        }

Result:
   $result = mysqli_query($conn, $sql);        | $pdo->exec($sql);
         if (mysqli_num_rows($result)>0){        $rows affected
Close:                  while($row = mysqli
   mysqli_close($conn);   _fetch_assoc($result)){
                           } $row['..'];
                                                 (1)
      Place holders ?
$sql= Insert into books values (?, ?)
   $statement = $pdo -> prepare($sql);
   $statement -> bindValue(1, $_POST['isbn']);
   $statement -> bindValue(2, $_POST['title']);
   $statement -> execute();
                      or array
  $statement = $pdo -> prepare ($sql);
  $statement -> execute(~~array~~(array($_Post['isbn'], $_Post['title']));
_______________________________________________________________
 values (:isbn, :title) :                         (2)
 $statement -> bind values(':title', $_POST['title']);
 $state -> execute(array(':isbn'=> $_POST['isbn'], ':title' => $_POST['title']
_______________________________________________________________
   transactions:
        try{
           $pdo-> query("");
           $pdo -> query("");
         $pdo -> commit();
       }catch (PDOEXCEPTION $e){
           $pdo -> rollback();
       }
```


CSS

$\overbrace{\text{property1: value1}}^{\text{declaration}}$

selector $\underbrace{\text{\{ property1: value ; property 2: value2; \}}}_{\text{declaration block}}$ ← rule

props:

font
font-family
font-size
font-style
font-weight
@font-face
letter-spacing
line-height
text-align
text-decoration
text-indent
background
background-color
background-image
background-position
background-repeat
color: / color: rgb(r,g,b);
rgba(r,g,b,a)
border
border-color
" -width
" -style
" -top
top-color
" top-width

px : absolute.
em : rel. to parent
rem : rel. to root
% : dep.
inch, cm, mm, pt, pc : abs.

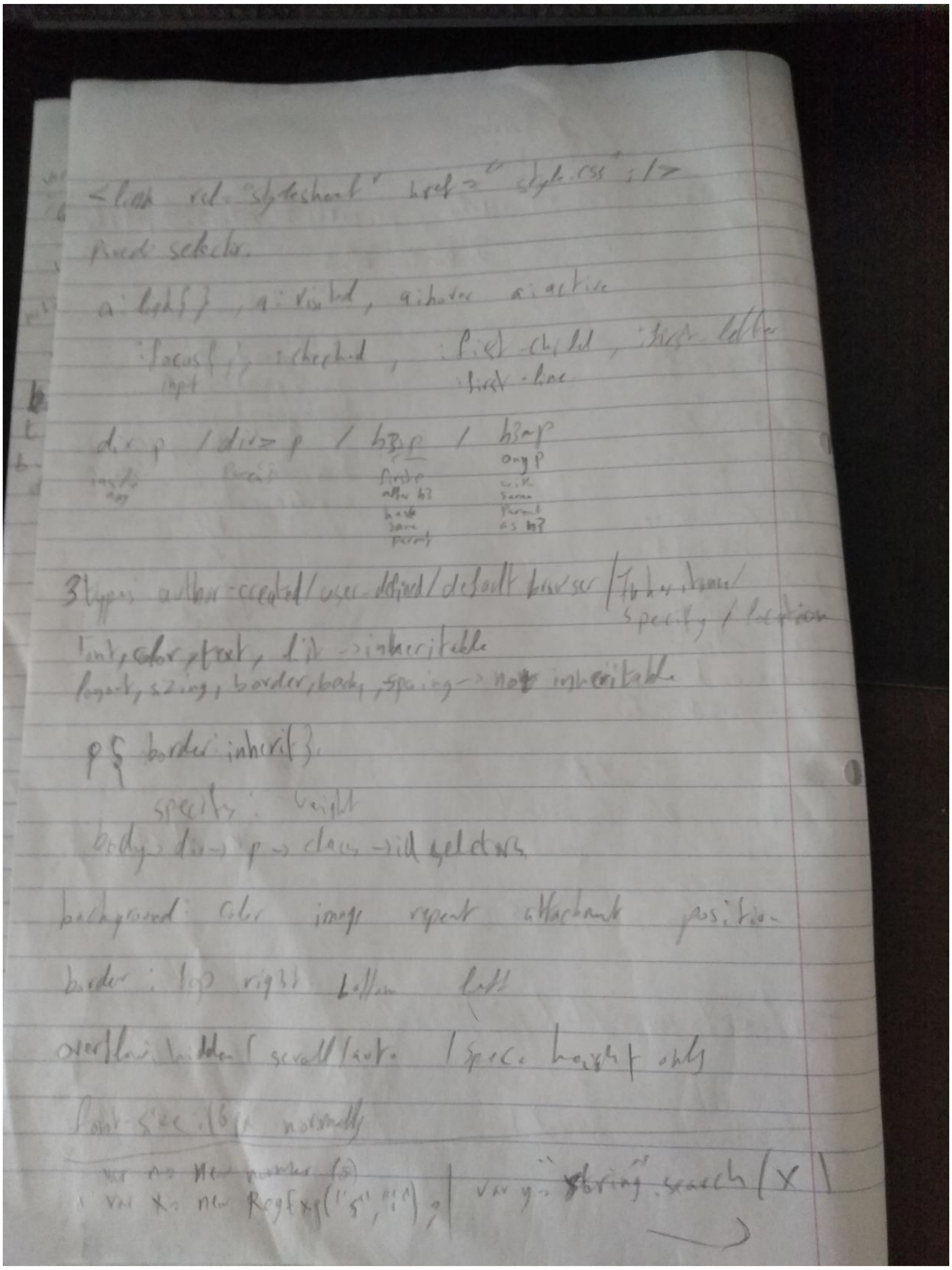
p	*/

* : all elements

Inline / embedded / External

class	id	att
.selector	#selector	[title] :

sel att value
a[title] = " "
{~ ^ ~} contains
start ~(^ =) [illegible]
substring ~(* =) [illegible]
end ~($ =)



<link rel="stylesheet" href="style.css" />

Pseudo selector.

a:link{} , a:visited, a:hover a:active

:focus{} , :checked , :first-child , :first-letter
input   :first-line

div p / div > p / h3+p / h3~p

inside any / parent / first p after h3 have same parent / any p with same parent as h3

3 types author-created / user-defined / default browser | Inheritance / Specificity / location

font, color, text, list -> inheritable
layout, sizing, border, back, spacing -> not inheritable

p{ border: inherit }.

specificity: weight
body -> div -> p -> class -> id selectors

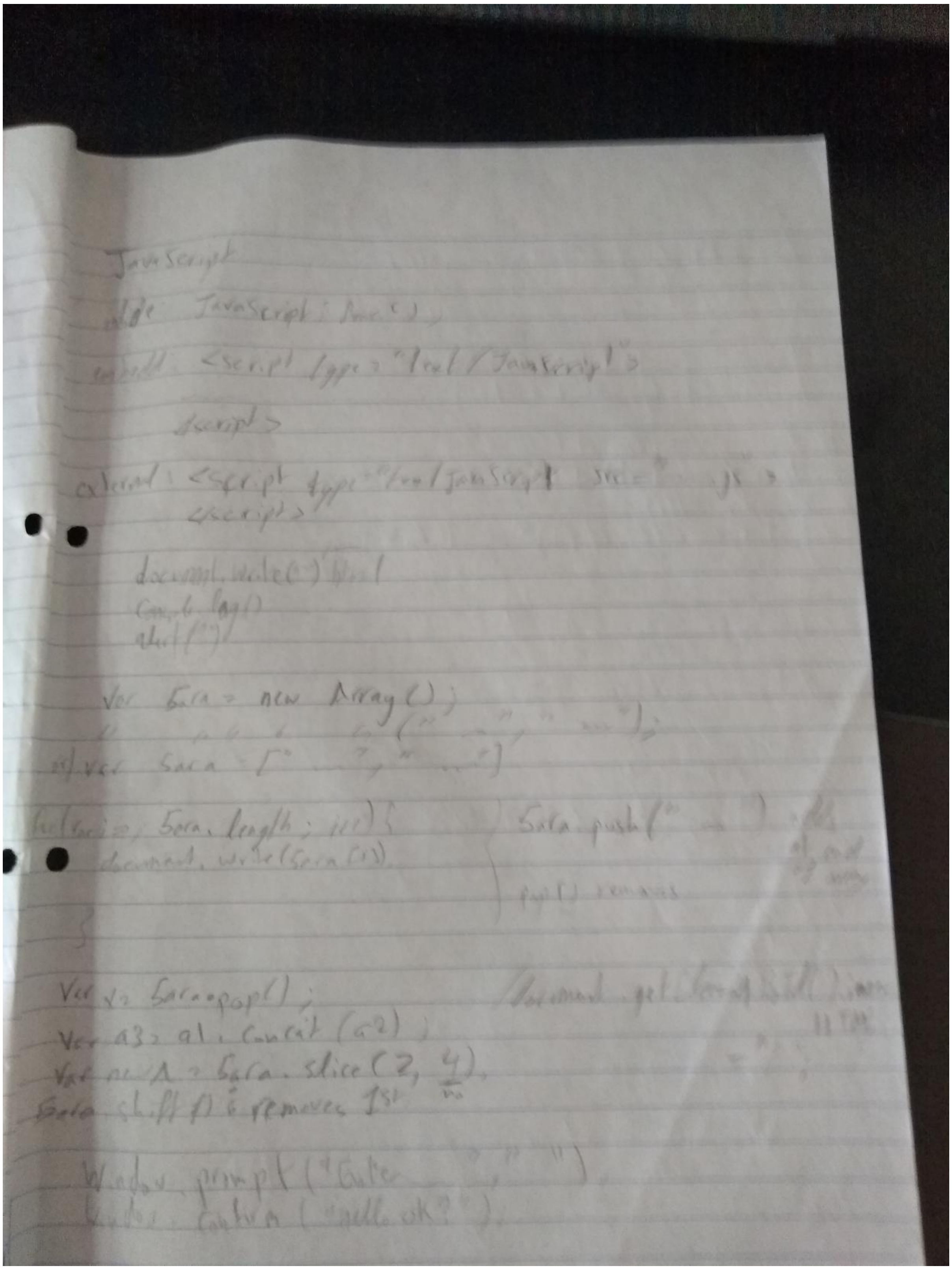
background: color image repeat attachment position

border: top right bottom left

overflow: hidden / scroll / auto. / spec. height only

font-size: 16px normally

~~var n = new Number (5)~~
var x = new RegExp("s", "i"); | var y = string.search(x)



JavaScript

inline JavaScript: [illegible]( )

internal: <script type="text/JavaScript">

</script>

external: <script type="text/JavaScript" src=" .js">
</script>

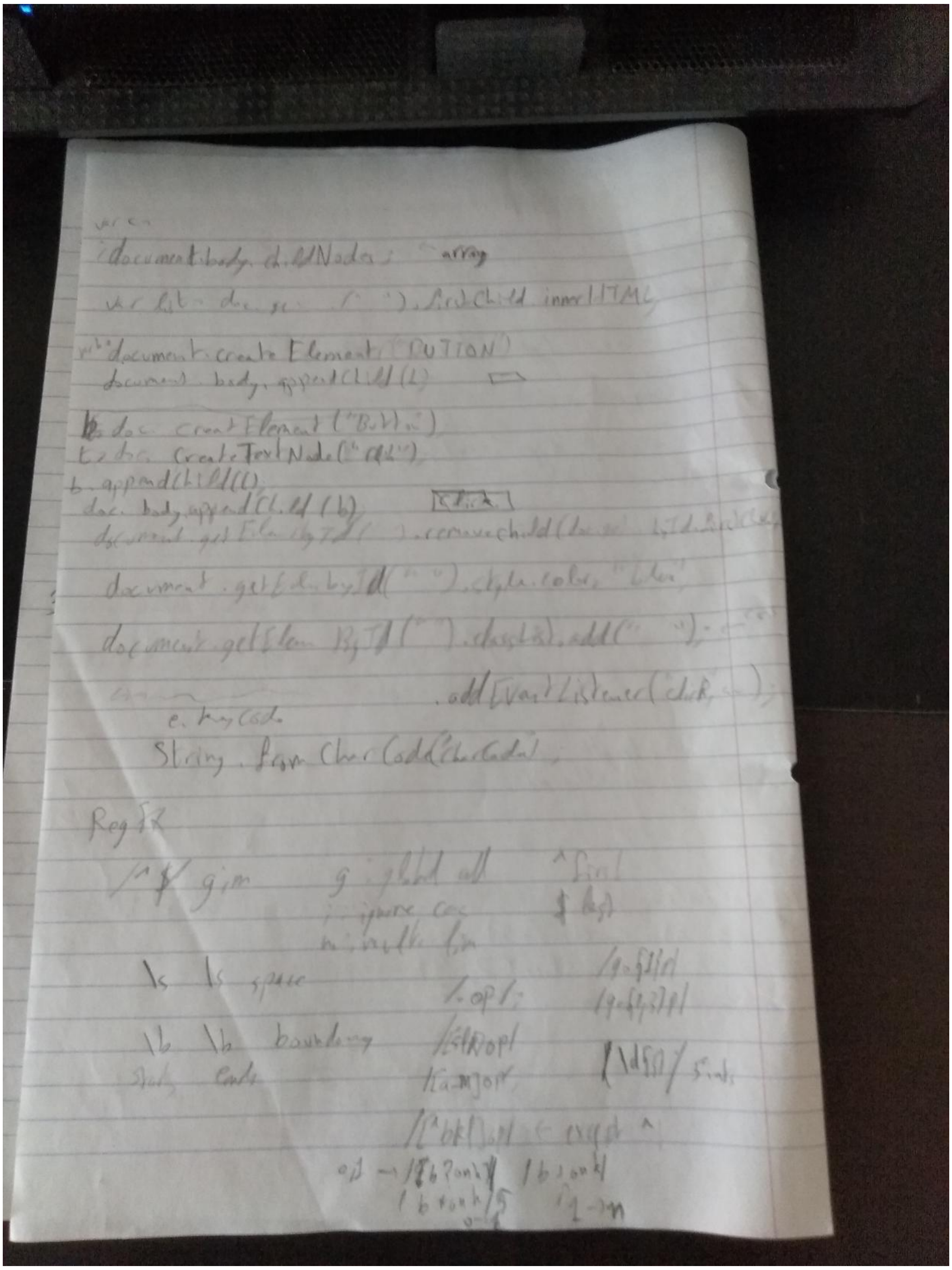
```
document.write(" ") html
console.log()
alert(" ")

var Sara = new Array( );
 "    "   "   "   (" ", " ", " ... ");
or var Sara = [" ", " ... "]

for(var i=0; Sara.length; i++){
  document.write(Sara[i]);

}
```

Sara.push(" ") [illegible] at end of array

pop() removes

document.getElementById( ).innerHTML
= " ";

```
var x = Sara.pop();
var a3 = a1.concat(a2);
var newA = Sara.slice(2, 4);
Sara.shift() removes 1st

Window.prompt("Enter ", " ")
Window.confirm("hello ok?");
```



```
var c=
document.body.childNodes;   array

var list= doc.qs   ("  ").firstChild innerHTML

var b=document.createElement("BUTTON")
 document.body.appendChild(b)        ▭

b=doc.createElement("Button")
t=doc.createTextNode("click")
b.appendChild(t)
 doc.body.appendChild(b)           [click]
 document.getElemByTd(  ).removeChild(document.getElemById.firstChild)

 document.getElemByTd("  ").style.color="blue"

 document.getElemByTd("  ").classList.add("  ");

 [illegible]                     .addEventListener(click,  );
      e.keyCode
    String.fromCharCode(charCode);

Reg Ex

  /^$/ gim        g global all        ^ first
                  i ignore case       $ last
                  m multiline

    \s  \S  space                     /go{3}d/
                          /.op/;      /go{4,3}d/

    \b  \B  boundary      /[tk]op/
   start  end             /[a-m]op/   /\d{5}/ 5 ints

                          /[^bk]op/ ← except ^
             op → /\b?ont/  /b?ont/
                  /b+ont/   1-m
```