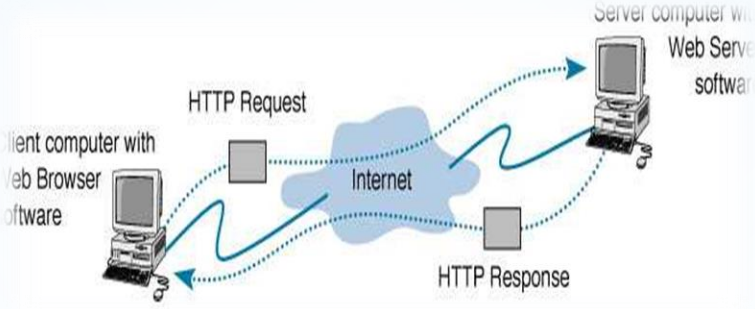
Dynamic Web Pages

Server Side Scripting



http://www.Madadyar.ir

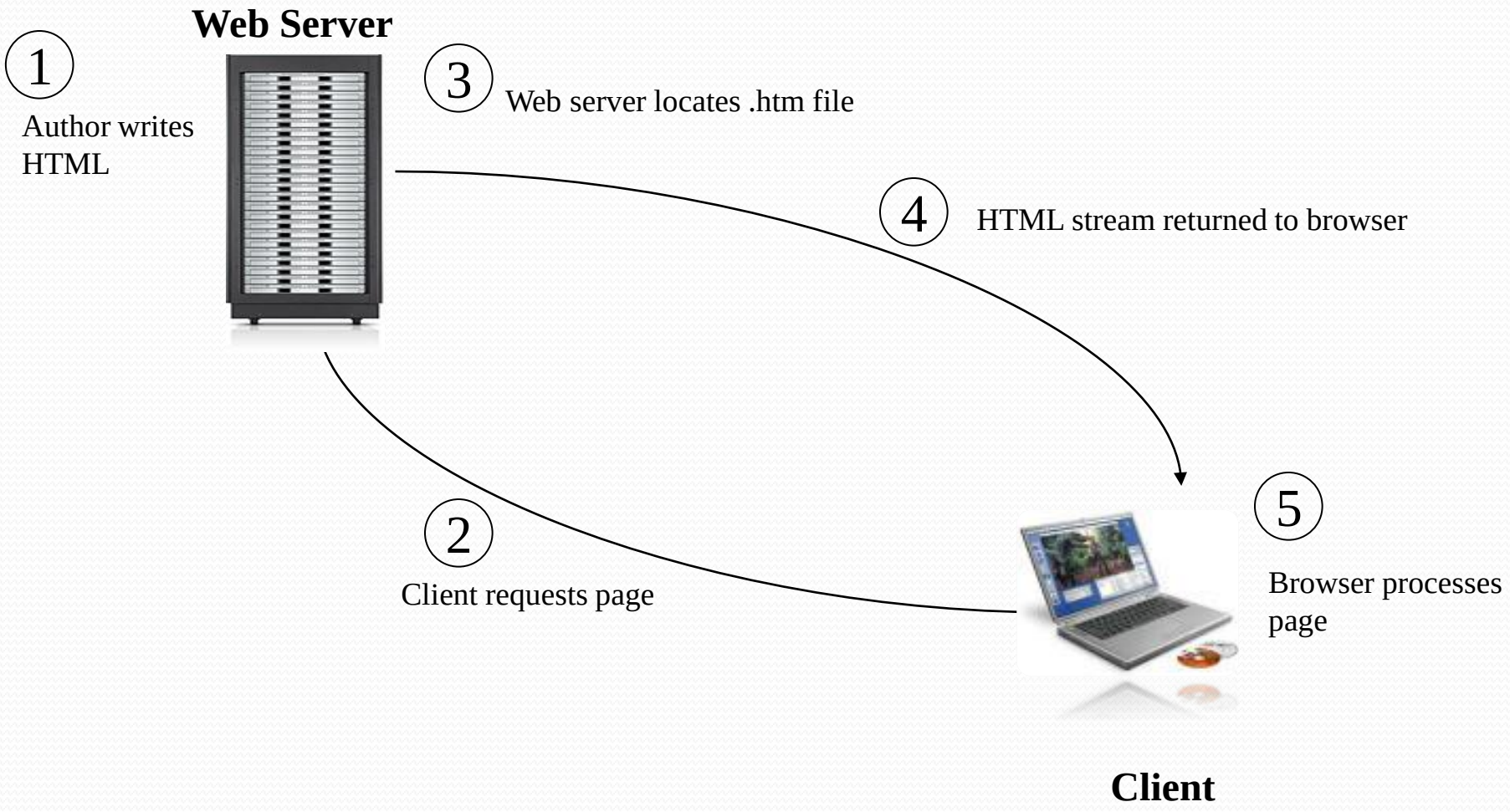
Advantage of Client/Server Architecture

- Easy to maintain.
- Security Control.
- Easy to update.
- Multiple different clients with different capabilities.

Static Web Pages

- Page content doesn't change.
  - Created With HTML/XHTML and CSS.

Static Web Page Delivery

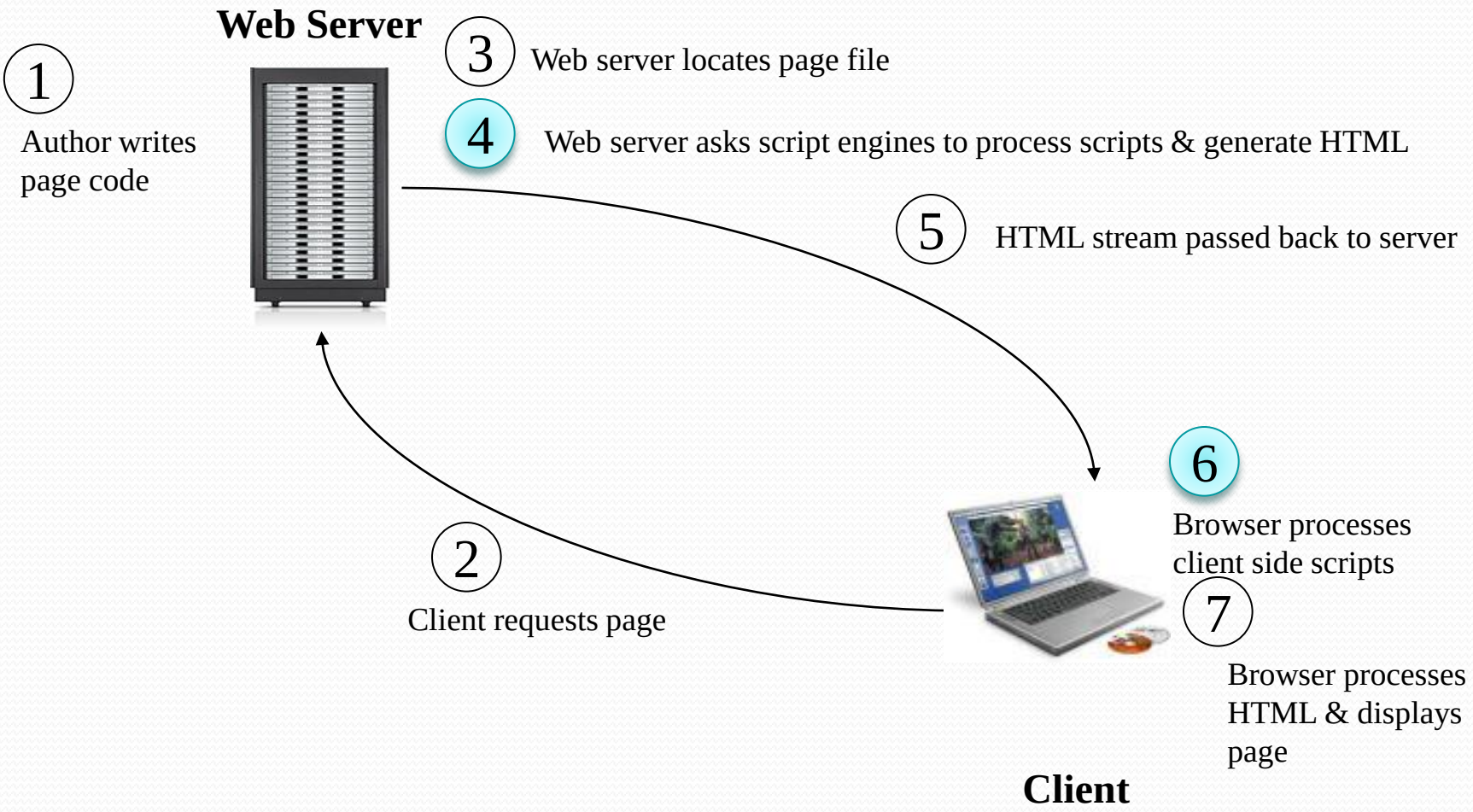


Dynamic Web Pages

- Created **on the fly** based on user actions or current conditions.

- Allows the page to be customized to the user and the situation.

Dynamic Web Pages Delivery



Server-side vs. Client-side Processing

- Computer processing can happen in two locations
  - **Server**:
    - Accepts request, finds page, generate Output[HTML] sends it.
  - **Client**:
    - Gets HTML from net, processes it, displays it.
- Advanced things can happen on one or both sides

Many Technology Choices

- **Client-Side Technologies**
  - Scripting languages: **JavaScript**, VBScript
- **Server-Side Alternatives**
  - CGI (Common Gateway Interface)
  - **Active Server Pages (ASP)/ASP.NET**
  - PHP
  - Java Server Pages (JSP)
  - Perl

Server Technologies

Name	Webserver	Operating System
ASP.NET	IIS	Windows
PHP	Apache	Linux/Windows
Java (JSP)	Tomcat	Linux/Windows

Server-side vs Client-side Scripts

Server-side

- Processed by webserver.
- Does not rely on browser support.
- Slower Run & more security
- Script code not visible in page source
- Can
  - Manage sessions (shopping baskets, etc.)
  - Database processing.

Client-side

- Processed by browser.
- Does not depend on web server requirements.
- Faster Run & less security
- Script code is viewable in page source.

What can be done with Server Pages

- Dynamically edit, change or add any content of a Web page.
- Respond to user queries or data submitted from HTML forms.
- Access any data or databases and return the results to a browser .

An Example With ASP.NET

```
<html>
<head></head>
<body>
<%
   Response.Write("hello, asp.net world!“);
%>
</body></html>
```

An Example With PHP

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

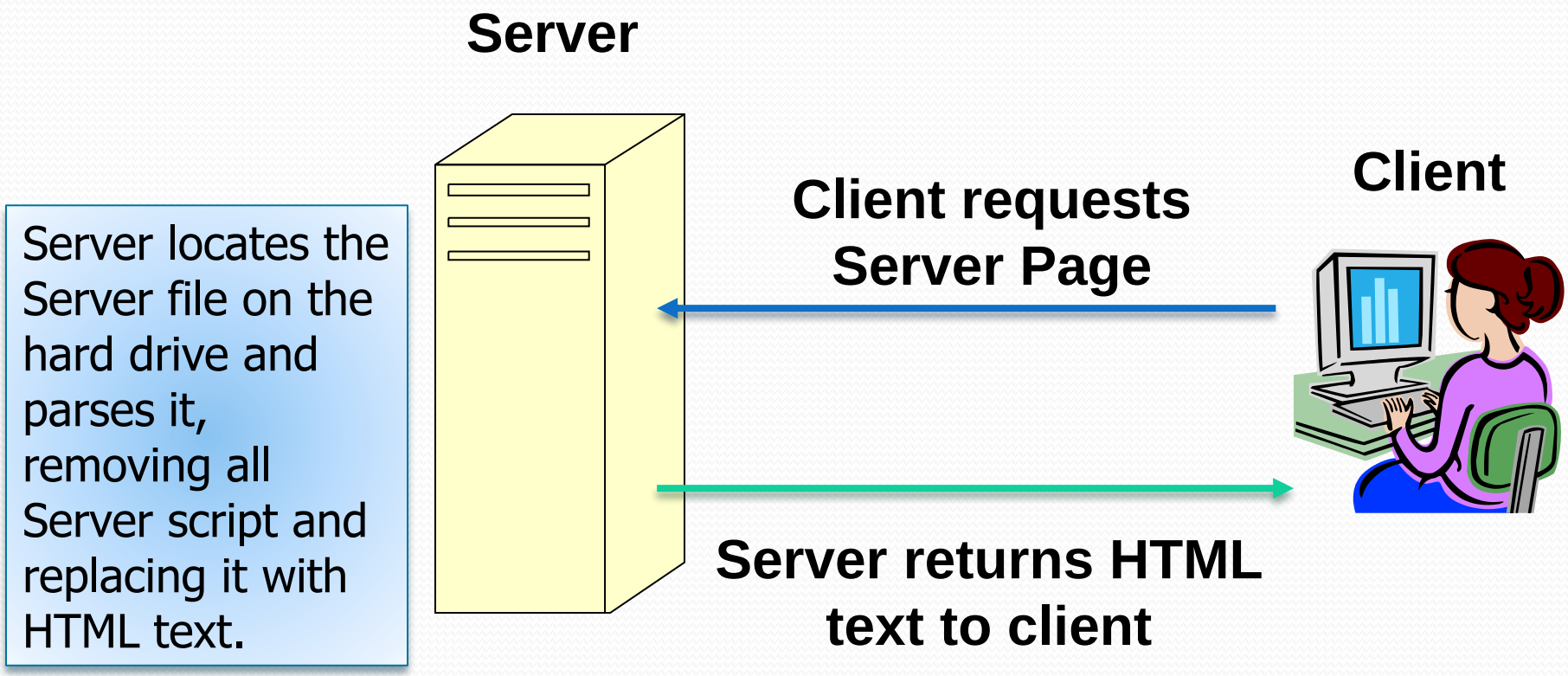
An Example With JAVA

```
<!DOCTYPE html>
<html>
<body>

<%= "My first JAVA script!"; %>

</body>
</html>
```

Client/Server Interaction for Server Pages



Client/Server Interaction for ASP.NET

File On Server

```
<html>
<head> <title>hello world</title>
</head><body>
<%
    // This will print to the browser the
    // words Hello, ASP World.
    Response.Write("<br/>Hello, ASP.NET World!");
%>
</body></html>
```

Result on Client

```
<html>
<head> <title>hello world</title>
</head><body>
<br/>
Hello, ASP.NET  World!
</body></html>
```

SERVER Objects

- **Request**
- **Response**
- **Server**
- **Application**
- **Session**

Request

- Can get input from query string or form.
- Can get cookie information.
- Can also get total bytes, certificate, ….
- Example:
  - <% Request.QueryString [“fname”] %>

Response

- Can send output to user through web page.
- Can set cookie values.
- Can set character set, expiration.
- Can clear, write output.
- Can redirect.
- Example:
  - <% Response.Write “message” %>

Session

- Store user information during request and response and request and response….
- Want to identify, maintain user information or state in stateless HTTP protocol.
- Client has an id number and expiration time from last request or expires.
- Client can terminate or abandon causing Session object to be destroyed as well.

Application

- Information about entire website.
- Global variables.
  - Example is a counter.
- Lock and Unlock to access to update.

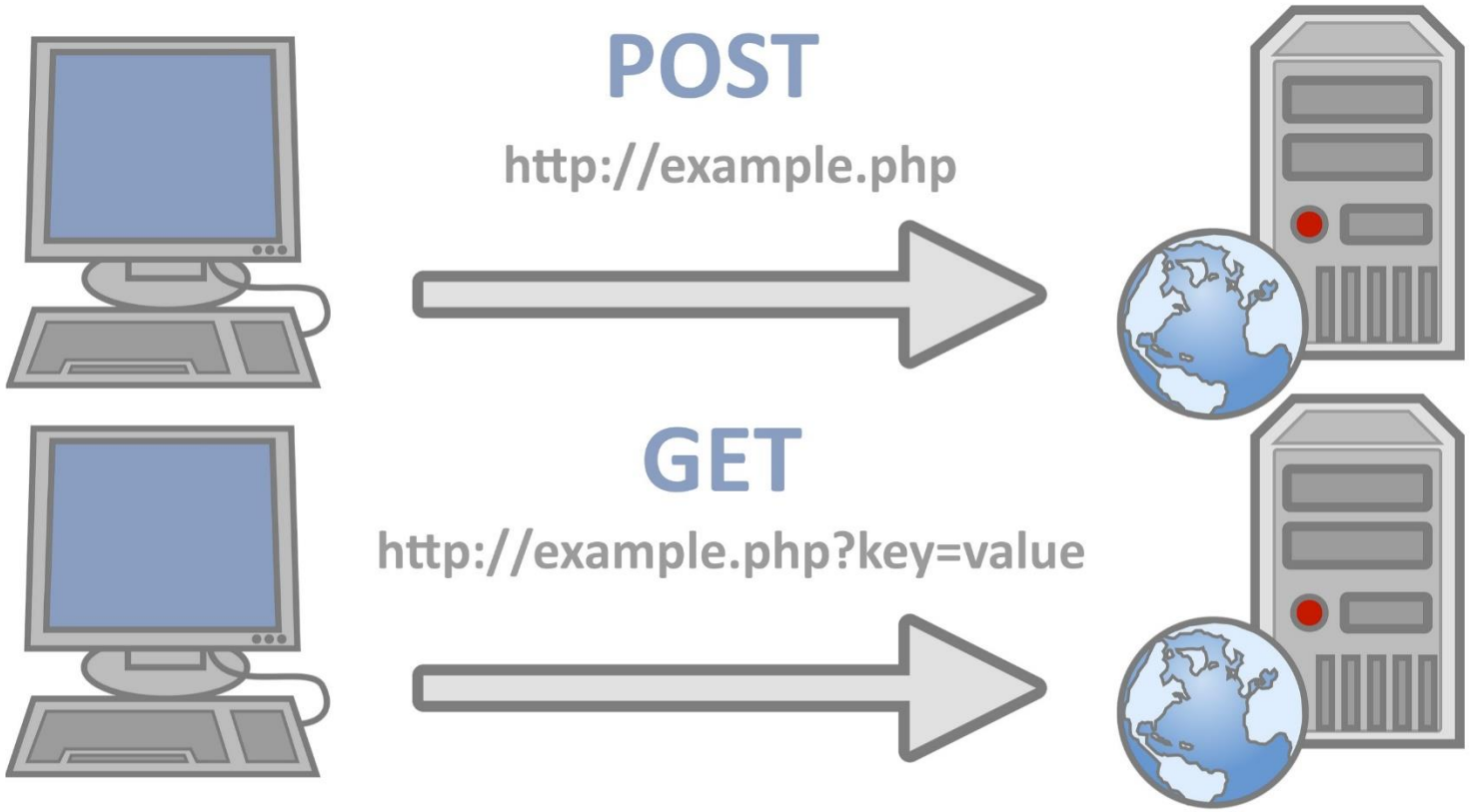
Server

- Server related utility functions.
- Create an object like DB object.
- Can enable URL encoding.
- Example:
  - Server.Mappath(Address of file)

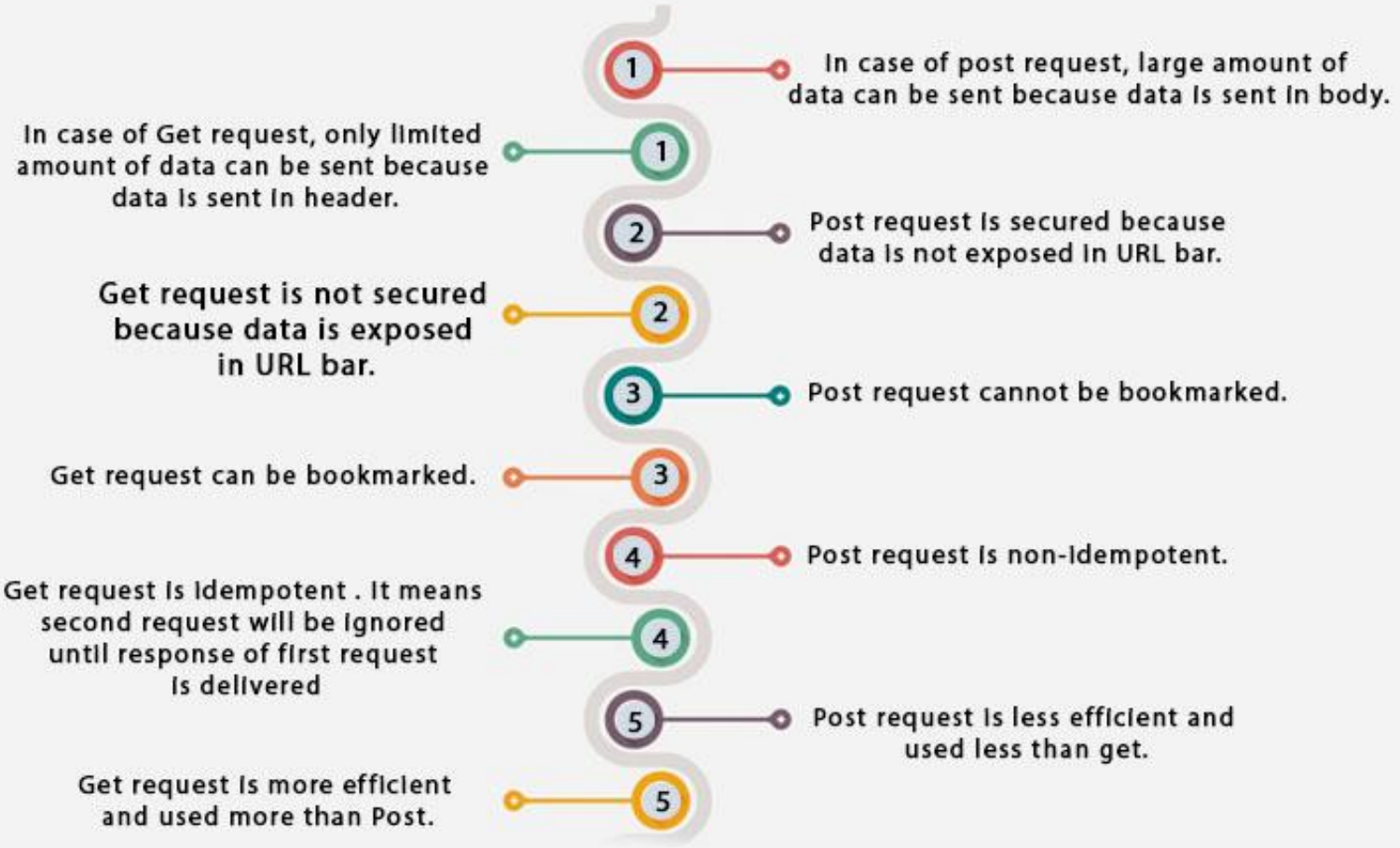
Cookies

- **Cookies** provide a means in Web applications to store user-specific information.
  - For example, when a user visits your site,
- you can use cookies to store user preferences or other information.
- When the user visits your Web site another time, the application can retrieve the information it stored earlier.

Send Data To Server



Get vs. Post


1
In case of post request, large amount of data can be sent because data is sent in body.
In case of Get request, only limited amount of data can be sent because data is sent in header.
1
2
Post request is secured because data is not exposed in URL bar.
Get request is not secured because data is exposed in URL bar.
2
3
Post request cannot be bookmarked.
Get request can be bookmarked.
3
4
Post request is non-idempotent.
Get request is idempotent . It means second request will be ignored until response of first request is delivered
4
5
Post request is less efficient and used less than get.
Get request is more efficient and used more than Post.
5