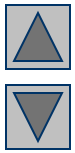
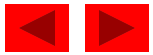


Outline

Server Side Web Technologies

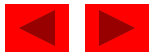
Client Side vs. Server Side Web

- Simply defined, client-side code executes on the end-user's computer, usually within a web browser.
- Server-side code executes on the web server, usually within a web application environment, which in turn generates HTML to be viewed in a browser.



Client Side vs. Server Side Web

- Which one to choose? What are the determining factors?
  - Performance:
    - Responsiveness, speed, reliability
    - Ability to handle a large number of simultaneous users
  - Functionality:
    - Simplicity of use and maintenance,
    - Breadth of user options
    - Ability to handle multiple simultaneous transactions
  - Security:
    - Desktop security
    - Server security
    - Database security
    - Network security

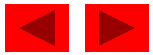


Client side scripting

- web browsersweb browsers execute client side scripting. It is use when browsers has all code. Source code used to transfer from web serverweb browsers execute client side scripting. It is use when browsers has all code. Source code used to transfer from web server to user's computer over internet and run directly on browsers. It is also used for validations and functionality for user events.
- It allows for more interactivity. It usually performs several actions without going to user. It cannot be basically used to connect to databases on web server. These scripts cannot access file system that resides at web browser. Pages are altered on basis of users choice. It can also used to create "cookies" that store data on user's computer.

Server side scripting :

- Web servers are used to execute server side scripting. They are basically used to create dynamic pages. It can also access the file system residing at web server. Server-side environment that runs on a scripting language is a web-server.
- Scripts can be written in any of a number of server-side scripting language available. It is used to retrieve and generate content for dynamic pages. It is used to require to download plugins. In this load times are generally faster than client-side scripting. When you need to store and retrieve information a database will be used to contain data. It can use huge resources of server. It reduces client-side computation overhead. Server sends pages to request of user/client.



Difference between client side scripting and server side scripting

Client side scripting	Server side scripting
Source code is visible to user.	Source code is not visible to user because it's output of server side is a HTML page.
It usually depends on browser and it's version.	In this any server side technology can be use and it does not depend on client.
It runs on user's computer.	It runs on web server.

There are many advantages link with this like faster.
response times, a more interactive application.

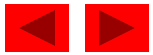
The primary advantage is it's ability to highly customize, response requirements, access rights based on user.

It does not provide security for data.

It provides more security for data.

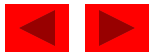
It is a technique use in web development in which scripts runs on clients browser.

It is a technique that uses scripts on web server to produce a response that is customized for each clients request.



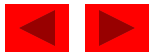
Client Side Technologies

- HTML : markup language for display of web content
  - DHTML extensions for dynamic and interactive control of web page content and display (Not fully standardized by W3C yet)
  - Tools for writing html documents include : Dreamveawer, FrontPage and any word processor (including Notepad)
- JavaScript: client side programming language
- VBScript: client side programming language (MS proprietary, supported by IE)



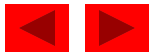
Client Side Technologies

- Java Applets:
  - small programs written in Java, embedded in an HTML page and executed from within a browser
  - Unlike JavaScript, the Java code must be pre-compiled into a so-called *bytecode* before it can be interpreted by a browser's so–called Java Virtual Machine
  - In other words, the Notepad and the browser alone are not enough to write java applets



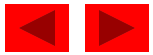
Client Side Technologies

- ActiveX controls
  - Similar to Java Applets but can be written in a variety of programming languages such as C, C++, VB and even Java
  - Supported by Windows only
  - Security issues: unlike Java applets, ActiveX controls have full access to all desktop resources: memory, operating systems, …
    - Authentication and registration system



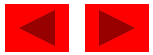
Client Side Technologies

- Macromedia Flash
  - Proprietary commercial application for creating interactive graphic content
  - It has its own scripting language
  - To reproduce the Flash content browsers must be equipped with a Flash Player plug-in



Client Side Technologies: summary

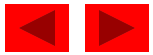
- Client Side technologies have evolved form a simple tools for creating static pages to sophisticated array of technologies turning a browser into a powerful multifunctional client
- Consequently, we can stop referring to a web client as “thin” client (i.e. limited in size and computational needs)



Server Side Technologies

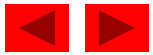
Server-side technologies are quite numerous and diverse. Popular server side web application technologies include:

- Microsoft ASP/.NET
- Java server technologies such as J2EE, JSP, and servlets
- CGI / Perl
- PHP
- ColdFusion



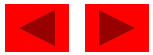
Server Side Technologies

- In addition, the server-side technologies include database systems such as Oracle, SQL Server (Microsoft), MySQL (open source) and many others
  - DB systems are indispensable part of server side operations and some DB software providers, such as Oracle are combining web application functionality with their core database functions



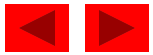
Server Side Technologies

- The “core” server side application development platforms can retrieve, modify and query the contents of databases through their own access mechanisms:
  - ADO.NET for Microsoft’s .NET platform enables access to almost every existing database platform
  - php enables direct access to many existing DB platforms, most notably MySQL, but also, Oracle, SQL Server and others

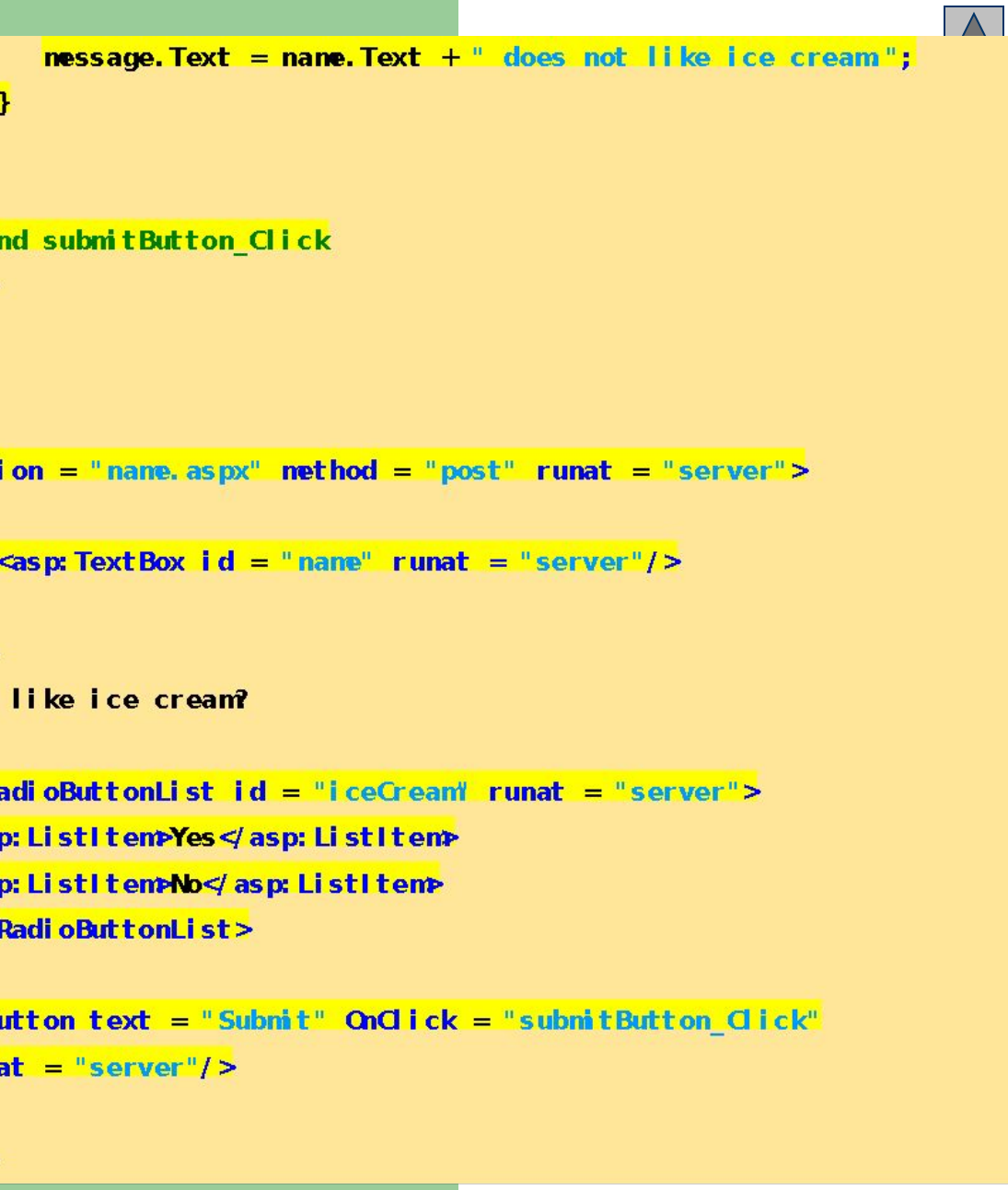


Server Side Technologies: ASP/.NET

- .NET is Microsoft framework supports many programming languages such as VB, C++, C#, JScript
  - One application can have components written in multiple languages
- ASP.NET (Active Server Pages) is an integral part of .NET initiative
  - It is a technology for creating dynamic web content on the server that appears as HTML on a client's browser
  - Developers can use this technology to write scripts in a language of their choice for from processing, interactive web pages, or any other dynamic content
- Every element in an ASP.NET page is treated as an object and run on the server.
- ASP.NET server controls are components that can perform the same work as HTML controls: radio buttons, text boxes, buttons, etc.
- Unlike HTML controls, ASP.NET controls preserve their content if and when this is needed

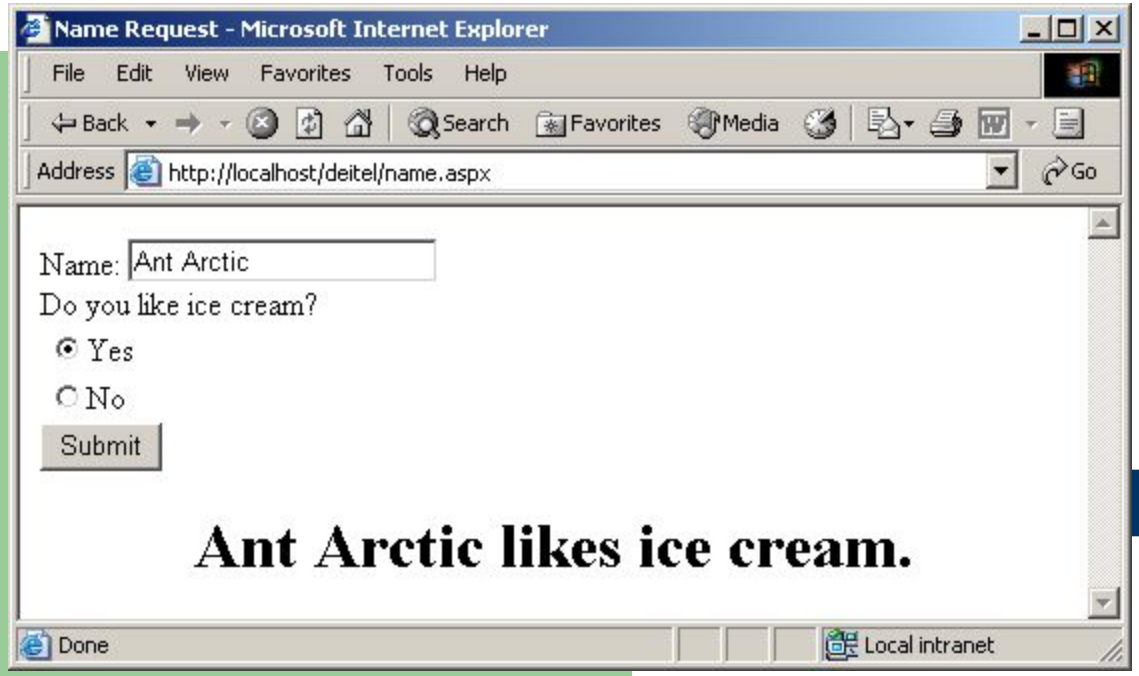


```
1  <%@ Page Language="JScript" %>
2
3  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
4    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
5
6  <!-- Fig. 23.9: name.aspx              -->
7  <!-- Another Simple ASP.NET example  -->
8
9  <html>
10    <head>
11       <title>Name Request</title>
12
13       <script language = "JScript" runat = "server">
14
15          function submitButton_Click(
16             sender : Object, events : EventArgs ) : void
17          {
18             if ( IsPostBack )
19             {
20                if ( iceCreamSelectedItem == "Yes" )
21                {
22                   message.Text = name.Text + " likes ice cream";
23                }
24                else
25                {
```



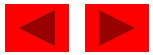
```
26                  message.Text = name.Text + " does not like ice cream";
27              }
28          }
29
30      } // end submitButton_Click
31  </script>
32  </head>
33
34  <body>
35      <form action = "name.aspx" method = "post" runat = "server">
36
37          Name: <asp:TextBox id = "name" runat = "server"/>
38
39          <br />
40          Do you like ice cream?
41
42          <asp:RadioButtonList id = "iceCream" runat = "server">
43              <asp:ListItem>Yes</asp:ListItem>
44              <asp:ListItem>No</asp:ListItem>
45          </asp:RadioButtonList>
46
47          <asp:Button text = "Submit" OnClick = "submitButton_Click"
48              runat = "server"/>
49
50          <br />
```

```
51          <center>
52             <h1> <asp:Label id = "message" runat = "server"/> </h1>
53          </center>
54
55       </form>
56    </body>
57 </html>
```

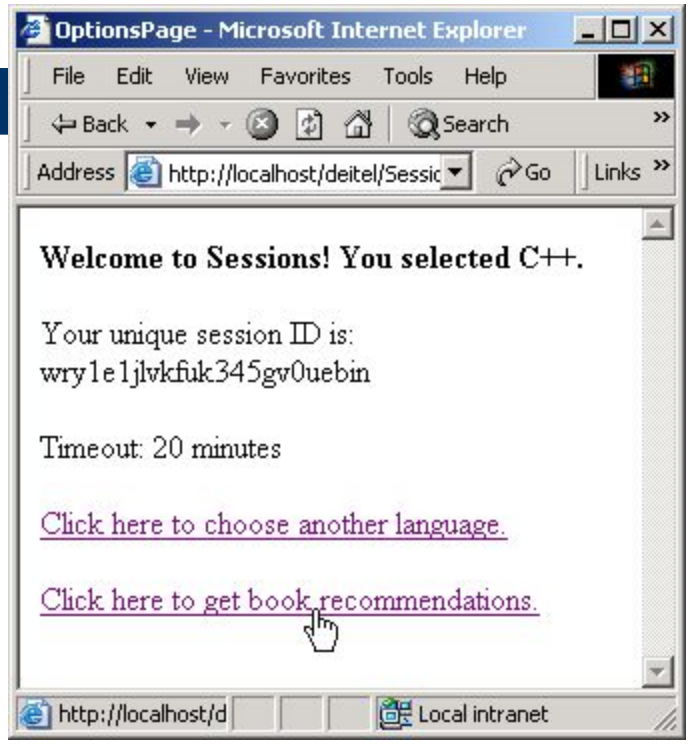


Server Side Technologies: ASP/.NET Session Tracking

- If the server (Web Host) running the server side script interacts with multiple clients (such as multiple customers buying goods at Amazon.com, for example)
- If interaction requires more than one http page request:
  - in essence more than one click of the button is needed to process a transaction
- The problems of session tracking is caused by the fact that the HTTP protocol is *stateless*
  - Every page load is a new event without memory of any previous events
- Not acceptable for any web application that is spread over the series of page loads, such as on-line shopping, catalog browsing, registration and large form entry pages, on-line questionnaires
- Two most common solutions
  - Cookies
  - Session Identifiers

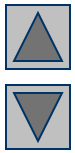



OptionsPage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search
Address http://localhost/deitel/Sessic Go Links
Select a programming language:
C#
C++
C
Python
Submit
Done
Local intranet


OptionsPage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search
Address http://localhost/deitel/Sessic Go Links
Welcome to Sessions! You selected C++.
Your unique session ID is:
wry1e1jlvkfuk345gv0uebin
Timeout: 20 minutes
Click here to choose another language.
Click here to get book recommendations.
http://localhost/d
Local intranet

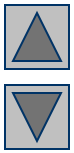
Outline

```
1  <%@ Page Language="JScript" %>
2  <%@ Import Namespace="System" %>
3
4  <%- Fig. 23.19: optionsPage.aspx                    --%>
5  <%- Page that presents a list of language options. --%>
6
7  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
8     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
9
10 <html>
11    <head>
12       <title>Options Page</title>
13
14       <script runat = "server">
15
16          // event handler for Load event
17          var books : Hashtable = new Hashtable();
18
19          function Page_Load( sender : Object, events : EventArgs ) : void
20          {
21             // if page is loaded due to postback, load session
22             // information, hide language options from user
23             books.Add( "C#", "0-13-062221-4" );
24             books.Add( "C++", "0-13-089571-7" );
25             books.Add( "C", "0-13-089572-5" );
```



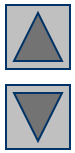
Outline

```
26          books.Add( "Python", "0-13-092361-3" );
27
28          if ( IsPostBack )
29          {
30             // display components that contain
31             // session information
32             welcomeLabel.Visible = true;
33             languageLink.Visible = true;
34             recommendationsLink.Visible = true;
35
36             // hide components
37             submitButton.Visible = false;
38             promptLabel.Visible = false;
39             languageList.Visible = false;
40
41             // set labels to display Session information
42             if ( languageList.SelectedItem != null )
43             {
44                welcomeLabel.Text +=
45                   languageList.SelectedItem.ToString() + ".";
46             }
47             else
48             {
49                welcomeLabel.Text += "no language.";
50             }
```



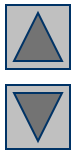
Outline

```
51
52          idLabel.Text += "Your unique session ID is: " +
53             Session.SessionID;
54
55          timeoutLabel.Text += "Timeout: " +
56             Session.Timeout + " minutes";
57       } // end if
58    } // end Page_Load
59
60    // when user clicks Submit button,
61    // store user's choice in session object
62    function submitButton_Click (
63       sender : Object, events : EventArgs ) : void
64    {
65       if ( languageList.SelectedItem != null )
66       {
67          var language : String =
68             languageList.SelectedItem.ToString();
69
70          // note: must use ToString method because the hash table
71          // stores information as objects
72          var ISBN : String = books[ language ].ToString();
73
74          // store in session as name-value pair
75          // name is language chosen, value is
```



Outline

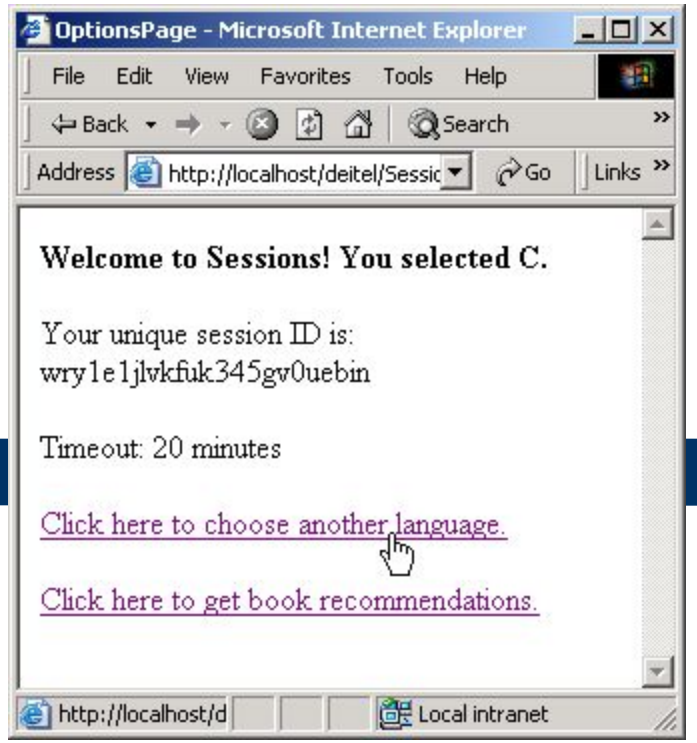
```
76              // ISBN number for corresponding book
77              Session.Add( language, ISBN );
78           } // end if
79        } // end submitButton_Click
80
81     </script>
82  </head>
83  <body>
84     <form id = "recommendationsPage" method = "post" runat = "server">
85        <P>
86        <asp:Label id = "promptLabel" runat = "server"
87           Font-Bold = "True">Select a programming language:
88        </asp:Label>
89        <asp:Label id = "welcomeLabel" runat = "server"
90           Font-Bold = "True" Visible = "False">
91           Welcome to Sessions! You selected
92        </asp:Label>
93        </P>
94        <P>
95        <asp:RadioButtonList id = "languageList" runat = "server">
96           <asp:ListItem Value = "C#">C#</asp:ListItem>
97           <asp:ListItem Value = "C++">C++</asp:ListItem>
98           <asp:ListItem Value = "C">C</asp:ListItem>
99           <asp:ListItem Value = "Python">Python</asp:ListItem>
100       </asp:RadioButtonList></P>
```



Outline

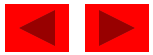
```
101         <P>
102            <asp:Button id = "submitButton" runat = "server"
103               Text = "Submit" onClick = "submitButton_Click">
104            </asp:Button>
105         </P>
106         <P>
107            <asp:Label id = "idLabel" runat = "server">
108            </asp:Label>
109         </P>
110         <P>
111            <asp:Label id = "timeoutLabel" runat = "server">
112            </asp:Label>
113         </P>
114         <P>
115            <asp:Label id = "newSessionLabel" runat = "server">
116            </asp:Label>
117         </P>
118         <P>
119            <asp:HyperLink id = "languageLink" runat = "server"
120               NavigateUrl = "optionsPage.aspx" Visible = "False">
121               Click here to choose another language.
122            </asp:HyperLink>
123         </P>
124         <P>
125            <asp:HyperLink id = "recommendationsLink" runat = "server"
```

```
126                NavigateUrl = "recommendationsPage.aspx"
127                Visible = "False">
128                Click here to get book recommendations.
129             </asp:HyperLink>
130          </P>
131       </form>
132    </body>
133 </html>
```



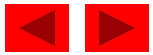
Web servers and HTTP (a primer)

- Web browsers communicate with web servers using the **H**yper**T**ext **T**ransfer **P**rotocol (HTTP). When you click a link on a web page, submit a form, or run a search, the browser sends an *HTTP Request* to the server.

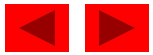


This request includes

- A URL identifying the target server and resource (e.g. an HTML file, a particular data point on the server, or a tool to run).
- A method that defines the required action (for example, to get a file or to save or update some data). The different methods/verbs and their associated actions are listed below:

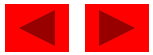


- `GET`: Get a specific resource (e.g. an HTML file containing information about a product, or a list of products).
- `POST`: Create a new resource (e.g. add a new article to a wiki, add a new contact to a database).
- `HEAD`: Get the metadata information about a specific resource without getting the body like `GET` would. You might for example use a `HEAD` request to find out the last time a resource was updated, and then only use the (more "expensive") `GET` request to download the resource if it has changed.
- `PUT`: Update an existing resource (or create a new one if it doesn't exist).
- `DELETE`: Delete the specified resource.
- `TRACE`, `OPTIONS`, `CONNECT`, `PATCH`: These verbs are for less common/advanced tasks, so we won't cover them here.



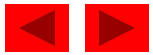
Web servers wait for client request messages, process them when they arrive, and reply to the web browser with an HTTP Response message. The response contains an HTTP Response status code indicating whether or not the request succeeded (e.g. "`200 OK`" for success, "`404 Not Found`" if the resource cannot be found, "`403 Forbidden`" if the user isn't authorized to see the resource, etc). The body of a successful response to a `GET` request would contain the requested resource.

When an HTML page is returned it is rendered by the web browser. As part of processing the browser may discover links to other resources (e.g. an HTML page usually references JavaScript and CSS pages), and will send separate HTTP Requests to download these files.



Compare GET vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached



History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL