Sending data: GET

<!DOCTYPE html>
<html>
<head>
  <title>Form</title>
</head>
<body>
  <form action="/response" method="GET">
    <input type="text" name="q">
    <input type="submit">
  </form>
</body>
</html>

GET /response?q=test HTTP/1.1

Click!
Client (browser)

Form displayed

User types “test” and clicks Submit

GET /form.html HTTP/1.1
Host: chunga.cs.unh.edu

HTTP/1.1 200 OK

<!DOCTYPE html>
<html>
  <form ...>
</html>

Server processes data and generates response.html

GET /response?q=test HTTP/1.1
Host: chunga.cs.unh.edu

HTTP/1.1 200 OK

<!DOCTYPE html>
<html>
  <p>User entered: test</p>
</html>
Sending data: POST

<!DOCTYPE html>
<html>
<head>
  <title>Form</title>
</head>
<body>
  <form action="/response" method="POST">
    <input type="text" name="q">
    <input type="submit">
  </form>
</body>
</html>

POST /response HTTP/1.1
Host: chunga.cs.unh.edu
Content-Type: application/x-www-form-urlencoded
Referer: http://chunga.cs.unh.edu/form.html
q=test
Sending data: POST

Client (browser)

GET /form.html HTTP/1.1
Host: chunga.cs.unh.edu

HTTP/1.1 200 OK

<!DOCTYPE html>
<html>
<form . . .
</form>
</html>

Web server

POST /response HTTP/1.1
Host: chunga.cs.unh.edu
q=test

HTTP/1.1 200 OK

<!DOCTYPE html>
<html>
<p>User entered: test</p>
</html>

Form displayed

User types “test” and clicks Submit

Server processes data and generates response.html
Not just forms: AJAX

```javascript
$("form").submit(function(f) {
    f.preventDefault();
    $.ajax({
        url: "/request",
        data: "q="+$("#q").val(),
        type: "GET",
        success: function(data) {
            var results = JSON.parse(data);
            $("#result").html(results["result"]);
        },
        error: function(e) {
            alert("Error: "+e.status+": "+e.statusText);
        }
    });
});
```

GET /request?q=test HTTP/1.1

{"result":"server response"}
HTTP transaction in general

Client

```
GET /request?param=value HTTP/1.1
...
```

— or —

```
POST /request HTTP/1.1
...
param=value
```

Server

```
HTTP/1.1 200 OK
...
Response
```

Server processes the request together with the values of the parameters and sends back the response.

HTML, JSON, XML, text, image, etc., specified by the Content-Type: line.
Server Architecture

- Infinite loop
  - while True do
    Accept Connection
    Process the Connection
done

- Concurrent request processing
  - while True do
    Accept Connection,
    have someone else to Process the Connection
done
Concurrent Requests

- Full-blown processes
  - costly in terms of resources
  - independent

- Threads
  - more lightweight
  - shared address space between requests

- Event-driven approach
  - do a quick bit of processing and schedule a callback when (typically) I/O is done
Event-driven approach

Node.js

```javascript
var http = require('http');

http.createServer((req, resp) => {
  resp.writeHead(200, {'Content-Type': 'text/plain'});
  resp.end('Hello World
');
}).listen(8000);

console.log('Server running at http://localhost:8000/');
```
Network Management

- Networks are complicated...

- Targets of management:
  - configuration
  - faults
  - performance
  - security
  - accounting
Network Management

- Two aspects of management
  - information collection and dissemination
  - decision making

- Components:
  - managed node
  - management station
  - management protocol
  - management information base (MIB)
Network Management

- Management station
- Router
- Switch
- Printer
- Host

Managed node
Management protocol
MIB
Management Protocols

- Simple Network Management Protocol (SNMP)
  - another “simple” protocol...
  - polling and trapping
  - data representation (ASN.1)
  - object identifiers (OIDs)

- OID Example
  - iso(1) identified-organization(3) dod(6) internet(1) mgmt(2) mib-2(1) ip(4) ipInReceives(3)
  - 1.3.6.1.2.1.4.3