Server side basics

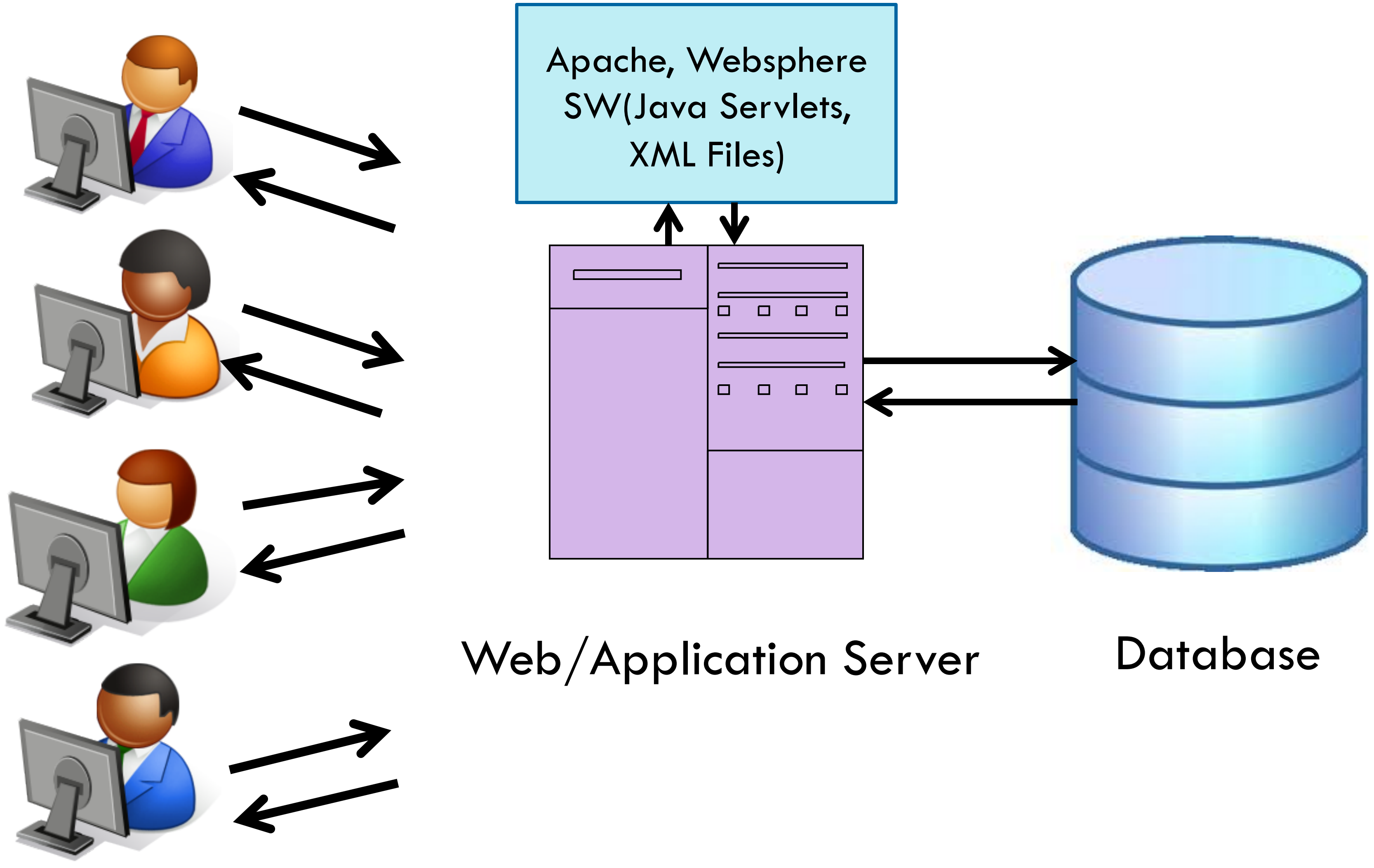
CSC309

URLs and web servers

```
http://server/path/file
```

- Usually when you type a URL in your browser:
  - Your computer looks up the server's IP address using DNS
  - Your browser connects to that IP address and requests the given file
  - The web server software (e.g. Apache) grabs that file from the server's local file system
  - The server sends back its contents to you

URLs and web servers (cont.)



URLs and web servers (cont.)

http://**www.facebook.com/home.php**

- Some URLs actually specify programs that the web server should *run*, and then send their output back to you as the result:
  - The above URL tells the server **facebook.com** to run the program **home.php** and send back its output

Server-Side web programming

- Server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby, ASP.NET, Python, Perl

Server-Side web programming (cont.)

- Dynamically edit, change or add any content to 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
- Customize a Web page to make it more useful for individual users
- Provide security since your server code cannot be viewed from a browser

Server-Side web programming (cont.)

- Web server:
  - contains software that allows it to run server side programs
  - sends back their output as responses to web requests
- Each language/framework has its pros and cons
  - we use Node.js in class

What is Node.js?

- Node.js is a platform that uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.
- Server-side scripting language
- Used to make web pages dynamic:
  - provide different content depending on context
  - interface with other services: database, e-mail, etc.
  - process form information

Why Node.js?

Blocking Code

```
var content = fs.readFileSync(filePath);
console.log( content);
console.log('Do somtheing else...');
```

Non-Blocking Code

```
fs.readFile( filePath, function (err, data) {
    console.log( content);
});
console.log('Do somtheing else...');
```

What App you should build using Node.js

Real-time applications:

online games,

collaboration tools,

chat rooms,

or robot? or sensor?

Basically, any application using "long-polling", you can write an application that sends updates to the user in real time.

Data Streaming

Where you should not use Node.js

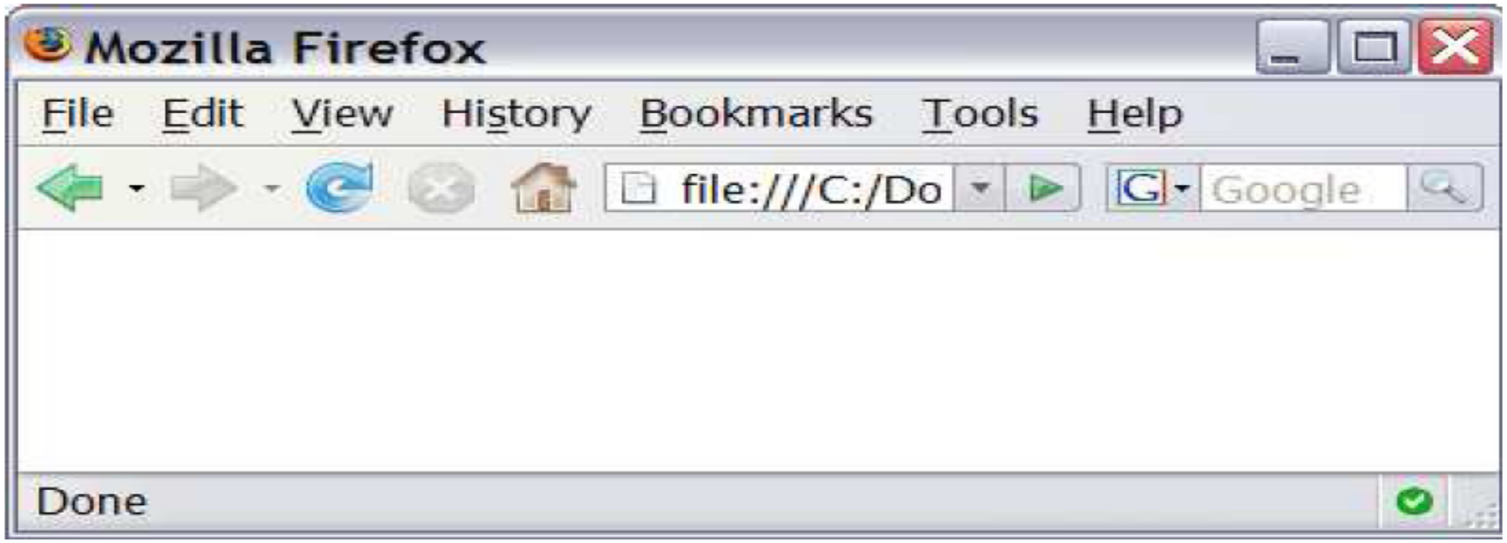
Computation heavy Server app

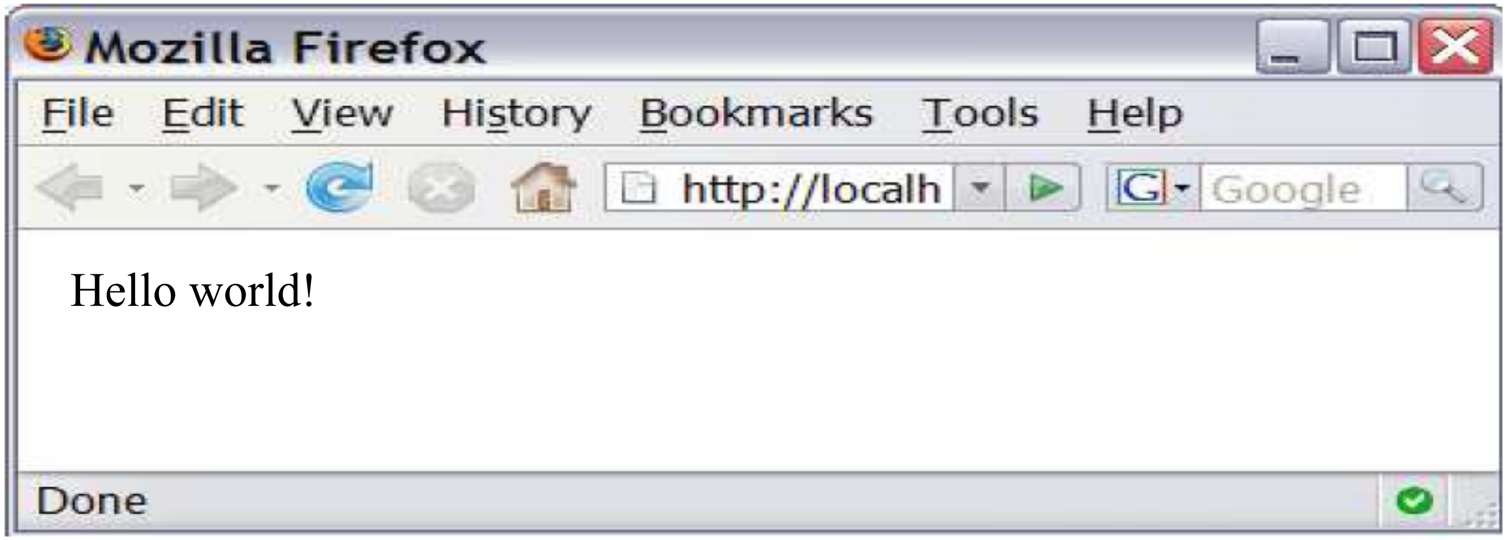
Hello World!

```
console.log('Hello World');
```

```
Hello world!
                                                            output
```

Viewing Node.js output





Hello world!

Node.js Basic Syntax

Node.js Syntax

Derived from JavaScript Syntax.

- Variables
- Arithmetic operators
- Comments
- Loops : for loop, while loop etc.
- Conditionals : if/else

Node.js Syntax

```
var http = require('http');
var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

UNIVERSITY OF TORONTO

Node.js Syntax

```
var http = require('http');
var server =
http.createServer(



);
server.listen(8080);
```

The server is just listening to the port 8080, doing nothing else.

Node.js Syntax

```
var http = require('http');
var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

Anonymous function

Node.js Syntax

```
var http = require('http');
var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

So, we are passing a function as a function parameter.

Node.js Syntax

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

function onRequest(req, res) {
  res.writeHead(200);
  res.end('Hello World');
}

http.createServer(onRequest).listen(80
80);
```

Node.js Syntax

Lets consider this generic example,

```
var result = database.query("SELECT *
FROM hugetable");
console.log("Hello World");
```

Node.js Syntax

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

HTTP Request and Response

```
var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

Node.js Syntax

```
var http = require('http'),
     fs = require('fs');

var server = http.createServer(function(req, res) {
    fs.readFile(__dirname + req.url, function (err,data) {
        if (err) {
          res.writeHead(404);
          res.end(JSON.stringify(err));
          return;
        }
        res.writeHead(200);
        res.end(data);
      });
});
server.listen(8080);
```