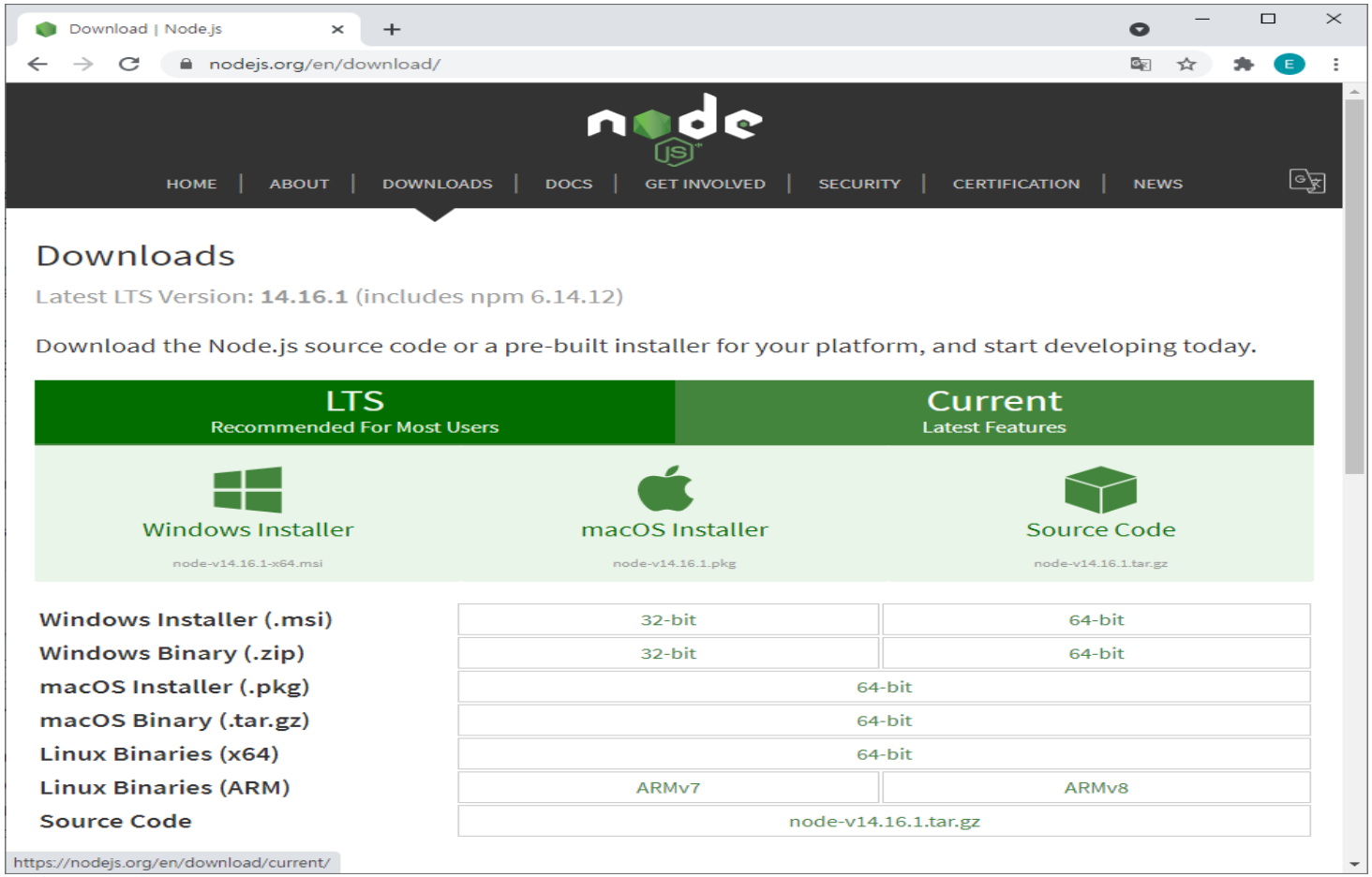
Node.js

Node.js

- **An open-source JavaScript runtime environment**
  - Built on Chrome's V8 JavaScript engine
  - It allows you to run **JavaScript on the server**
  - It runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)

- **Node JS applications:**
  - Netflix
  - LinkedIn
  - Wal-Mart
  - Paypal
  - YouTube
  - Amazone.com
  - eBay
  - Reddit
  - …

Node.js 설치


Download | Node.js
nodejs.org/en/download/
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS
Downloads
Latest LTS Version: 14.16.1 (includes npm 6.14.12)
Download the Node.js source code or a pre-built installer for your platform, and start developing today.
LTS
Recommended For Most Users
Current
Latest Features
Windows Installer
node-v14.16.1-x64.msi
macOS Installer
node-v14.16.1.pkg
Source Code
node-v14.16.1.tar.gz
Windows Installer (.msi) 32-bit 64-bit
Windows Binary (.zip) 32-bit 64-bit
macOS Installer (.pkg) 64-bit
macOS Binary (.tar.gz) 64-bit
Linux Binaries (x64) 64-bit
Linux Binaries (ARM) ARMv7 ARMv8
Source Code node-v14.16.1.tar.gz
https://nodejs.org/en/download/current/

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\user\Desktop\node01> node -v
v14.15.4
PS C:\Users\user\Desktop\node01> npm -v
7.9.0
PS C:\Users\user\Desktop\node01>
```

Creating Server

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

= http 모듈을 include 해라

```
http.createServer(function (request, response) {
```

서버에서 클라이언트로 보내는 것

```
   // Send the HTTP header
   // HTTP Status: 200 : OK
   // Content Type: text/plain
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Send the response body as "Hello World"
   response.end('Hello World₩n');
}).listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

Modules

- **Module**
  - Consider modules to be the same as JavaScript **libraries**.
  - A **set of functions** you want to include in your application.
- **Built-in Modules**
  - os
  - url
  - Query String
  - util
  - crypto
  - File system
  - http ~ 가장 기본적인
- **Include Modules**
  - Use the require() function with the name of the module:

HTTP Module

- **A built-in HTTP module**
  - It allows Node.js to transfer data over the Hyper Text Transfer Protocol
  - It can create an HTTP server that listens to server ports and gives a response back to the client.

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

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

Add an HTTP Header

- If the response from the HTTP server is supposed to be displayed as HTML:

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

첫번째인자 : statusCode (ex.404)

Create Your Own Modules:

- Use the **exports** keyword to make properties and methods available outside the module file.

```
exports.myDateTime = function () {
  return Date();
};
```

- Include Your Own Module

파일이름

```
var http = require('http');
var dt = require('./myfirstmodule');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```

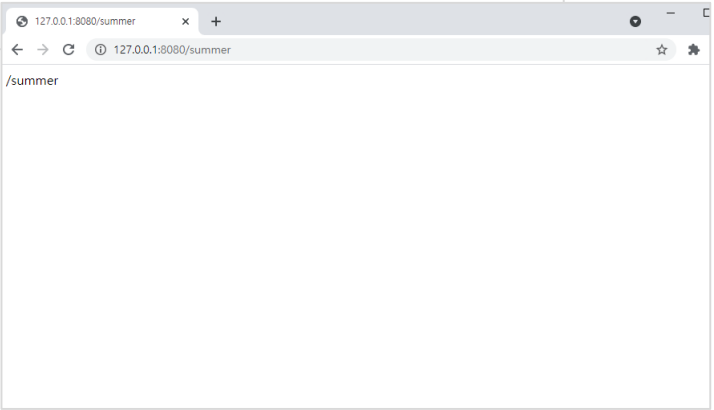
Read the Query String

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end();
}).listen(8080);
```

클라이언트에서 오는 요청객체

서버에서 보내는 객체

http://localhost:8080/summer



URL Module

- It splits up a web address into readable parts.
- Use **url.parse()** method

```
var url = require('url');
var adr = 'http://localhost:8080/default.htm?year=2017&month=february';
var q = url.parse(adr, true);

console.log(q.host); //returns 'localhost:8080'
console.log(q.pathname); //returns '/default.htm'
console.log(q.search); //returns '?year=2017&month=february'

var qdata = q.query; //returns an object: { year: 2017, month: 'february' }
console.log(qdata.month); //returns 'february'
```

객체

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var q = url.parse(req.url, true).query;
  var txt = q.year + " " + q.month;
  res.end(txt);
}).listen(8080);
```

http://localhost:8080/?year=2017&month=July

File System Module

- It allows you to work with the file system on your computer
  - Read / Create / Update / Delete / Rename files

- **Read Files : fs.readFile()**

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('demofile1.html', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

```
<html>
<body>
<h1>My Header</h1>
<p>My paragraph.</p>
</body>
</html>
```

↳ demofile1.html

Create Files

- **fs.appendFile()**
- **fs.open()**
- **fs.writeFile()**

파일을
없으면 만들고 추가
있으면 추가

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

fs.appendFile('mynewfile1.txt', 'Hello content!', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

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

fs.open('mynewfile2.txt', 'w', function (err, file) {
  if (err) throw err;
  console.log('Saved!');
});
```

write

열기모드

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

fs.writeFile('mynewfile3.txt', 'Hello content!', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

↳ 들어갈 내용

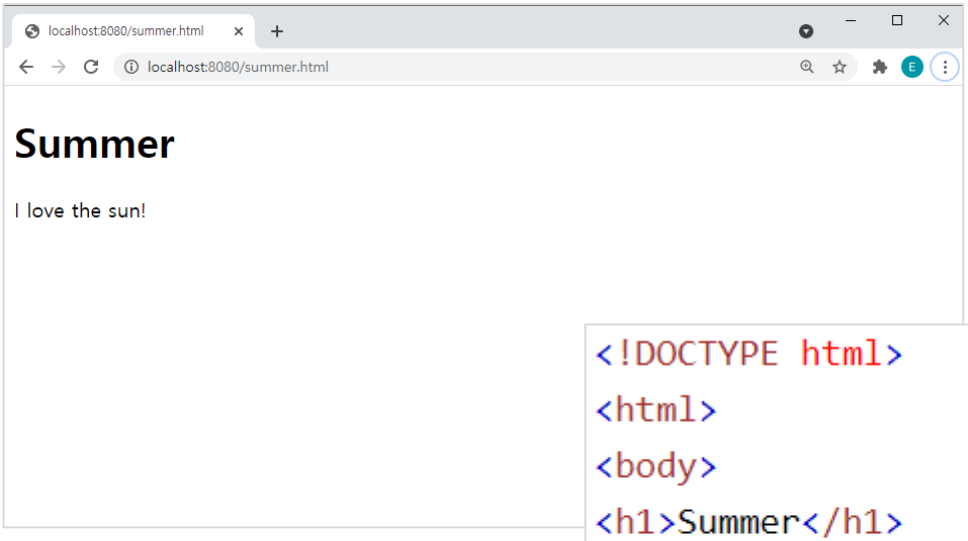
```
var fs = require("fs");

fs.writeFile('input.txt', 'Simply Easy Learning!', function(err) {
    if (err) {
        return console.error(err);
    }
    console.log("Data written successfully!");

    fs.readFile('input.txt', function (err, data) {
        if (err) {
            return console.error(err);
        }
        console.log(data.toString());
    });
});
```

없으면 다른 값이 나옴

Ex) File Server

http://localhost:8080/summer.html

http://localhost:8080/winter.html




```html
<!DOCTYPE html>
<html>
<body>
<h1>Summer</h1>
<p>I love the sun!</p>
</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>
<h1>Winter</h1>
<p>I love the snow!</p>
</body>
</html>
```

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

http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

http://localhost:8080/summer.html

http://localhost:8080/winter.html

return ./summer.html / ./winter.html

Delete Files

- **fs.unlink()**

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

fs.unlink('mynewfile2.txt', function (err) {
  if (err) throw err;
  console.log('File deleted!');
});
```

Rename Files

- **fs.rename()**

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

fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) {
  if (err) throw err;
  console.log('File Renamed!');
});
```

Get File Information

- **fs.stat(path, callback)**

```
var fs = require("fs");

console.log("Going to get file info!");
fs.stat('input.txt', function (err, stats) {
   if (err) {
      return console.error(err);
   }
   console.log(stats);
   console.log("Got file info successfully!");

   // Check file type
   console.log("isFile ? " + stats.isFile());
   console.log("isDirectory ? " + stats.isDirectory());
});
```

↳ status정보

true

false

Callback

인자

- **A function** passed **as an argument to another function.**
  - It allows a function to call another function.

- A callback function can **run after** another function has finished.

  request가 갈 때 실행됨

- **Node makes heavy use of callbacks.**
  - All the APIs are written in such a way that they support callbacks.

NoCallback.js

```
var fs = require("fs");
var data = fs.readFileSync('input.txt');

console.log(data.toString());
console.log("Program Ended");
```

실행결과

Simply Easy Learning (input.txt파일내용)
Program Ended

Callback.js

```
var fs = require("fs");

fs.readFile('input.txt', function (err, data) {
   if (err) return console.error(err);
   console.log(data.toString());
});

console.log("Program Ended");
```

비동기처리로 실행됨

↳ Callback함수

파일을 read하는 동안
기다리고 프로그램은 계속 실행되어
'Program Ended'가 먼저 실행됨

결과

Program Ended

Simply Easy Learning (input.txt 파일내용)

Events

- Node.js is a **single-threaded** application, but it can support concurrency via the concept of **event** and **callbacks**.

- Node.js uses **events** heavily and it is also one of the reasons why Node.js is pretty **fast** compared to other similar technologies.

```
var fs = require('fs');
var rs = fs.createReadStream('./demofile.txt');
rs.on('open', function () {
  console.log('The file is open');
});
```

파일을 읽을 수 있음

이벤트 연결 메소드

Events Module

사용자 정의 이벤트

- Node. js allows us to create and handle **custom events** easily by using **events** module
  - 1) use the require() method and create an EventEmitter object:

생성 객체

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

  - 2) bind an event handler with an event

```
// Bind event and event  handler as follows
eventEmitter.on('eventName', eventHandler);
```

↳해당 이벤트 발생시 일어날 동작에 대한 함수

  - 3) fire an event

```
// Fire an event
eventEmitter.emit('eventName');
```

```
var events = require('events');
var eventEmitter = new events.EventEmitter();

//Create an event handler:
var myEventHandler = function () {
  console.log('I hear a scream!');
}

//Assign the event handler to an event:
eventEmitter.on('scream', myEventHandler);
```

이벤트연결메소드

```
//Fire the 'scream' event:
eventEmitter.emit('scream');
```

이벤트 발생 메소드

```
var events = require('events');
var eventEmitter = new events.EventEmitter();

var connectHandler = function connected() {
   console.log('connection succesful.');③
   eventEmitter.emit('data_received');④
}

eventEmitter.on('connection', connectHandler);②
eventEmitter.on('data_received', function() {
   console.log('data received succesfully.');⑤
});

eventEmitter.emit('connection'); 이벤트발생됨

console.log("Program Ended.");
```

순서
①

NPM

- **Node Package Manager**
  - Command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

- **Installing Modules**

  $ npm install <Module Name>

- **Global vs Local Installation**

  $ npm install express

  $ npm install express –g

  전역설치 (다른프로젝트에서 별도 설치안해도됨)

- To check all the modules installed globally :

  $ npm ls –g

- **Ex)**

  $npm install upper-case

```
var http = require('http');
var uc = require('upper-case');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  /*Use our upper-case module to upper case a string:*/
  res.write(uc.upperCase("Hello World!"));
  res.end();
}).listen(8081);

console.log('Server running at http://127.0.0.1:8081/');
```

결과) HELLO WORLD

Express Framework

Express.js

- A minimal and flexible **Node.js web application framework**

- **Core features :**
  - Allows to **set up middlewares** to respond to HTTP Requests.
  - **Defines a routing table** which is used to perform different actions based on HTTP Method and URL.
  - Allows to **dynamically render HTML Pages** based on passing arguments to templates.

Running App

- Starts a server and listens on port 3000

```
var express = require('express');
var app = express();

app.listen(3000, function() {
   console.log("App listening on port 3000...")
})
```

앞의 http보다 간단해짐

Handling Requests with Express

- Express allows greater flexibility in responding to browser **'get' or 'post' requests.**
  ↳기본,default
  ex↳ form태그정보

- **Routing** refers to how an application's endpoints (URIs) respond to client requests

  ↱get, post

- **Basic routing :** app.METHOD(PATH, HANDLER)
  - app is an instance of express.
  - METHOD is an HTTP request method, in lowercase.
  - PATH is a path on the server.
  - HANDLER is the function executed when the route is matched.

```
app.get('/', function (req, res) {
   // --
})
```

라우터

callback function

callback function

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
   res.send('Hello World');
})

app.get('/about', function (req, res) {
   res.json({
      name : 'Greg Lim'
   });
})

app.listen(3000, function() {
   console.log("App listening on port 3000...")
})
```

Route Methods

```
var express = require('express');
var app = express();
// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
   console.log("Got a GET request for the homepage");
   res.send('Hello GET');
})

// This responds a POST request for the homepage
app.post('/', function (req, res) {
   console.log("Got a POST request for the homepage");
   res.send('Hello POST');
})

// This responds a DELETE request for the /del_user page.
app.delete('/del_user', function (req, res) {
   console.log("Got a DELETE request for /del_user");
   res.send('Hello DELETE');
})

// This responds a GET request for the /list_user page.
app.get('/list_user', function (req, res) {
   console.log("Got a GET request for /list_user");
   res.send('Page Listing');
})

// This responds a GET request for abcd, abxcd, ab123cd, and so on
app.get('/ab*cd', function(req, res) {
   console.log("Got a GET request for /ab*cd");
   res.send('Page Pattern Match');
})

app.listen(3000, function() {
   console.log("App listening on port 3000...")
})
```

어느것이 들어가도 상관x부분

- **app.all()**
  - Executed for requests to the route “/secret” whether using GET, POST, PUT, DELETE, or any other HTTP request method

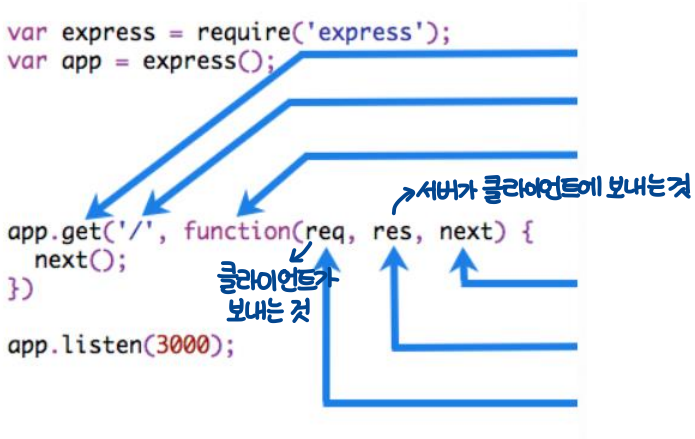
방식에 관계없이 동작함

```
app.all('/secret', function (req, res, next) {
  console.log('Accessing the secret section ...')
  next() // pass control to the next handler
})
```

Middleware

- **Middleware functions**
  - functions that have access to the request object (**req**), the response object (**res**), and the **next** middleware function

- It can perform the following tasks:
  - Execute any code.
  - Make changes to the request and the response objects.
  - Call the next middleware function in the stack.



- **app.use()**
  - To load the middleware function

```
var express = require('express')
var app = express()
     ↱ middleware function
var myLogger = function (req, res, next) {
  console.log('LOGGED')
  next()
}

app.use(myLogger)

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(3000)
```

Serving Static Files

- A built-in middleware **express.static** to serve static files

- if you keep your images, CSS, and JavaScript files in a public directory…

```
var express = require('express');
var app = express();

app.use(express.static('public'));

app.get('/', function (req, res) {
   res.send('Hello World');
})

app.listen(3000, function() {
    console.log("App listening on port 3000...")
})
```

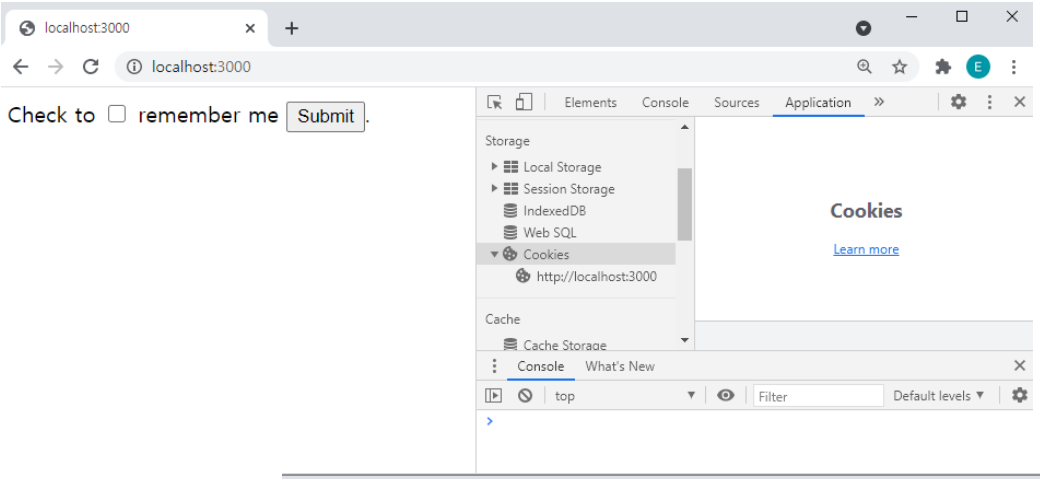
'public'이라는 폴더를 만들고
그 안에 'pic.jpg'라는 파일을 넣으면
'localhost:3000/pic.jpg'로 그 파일에
접근가능하다

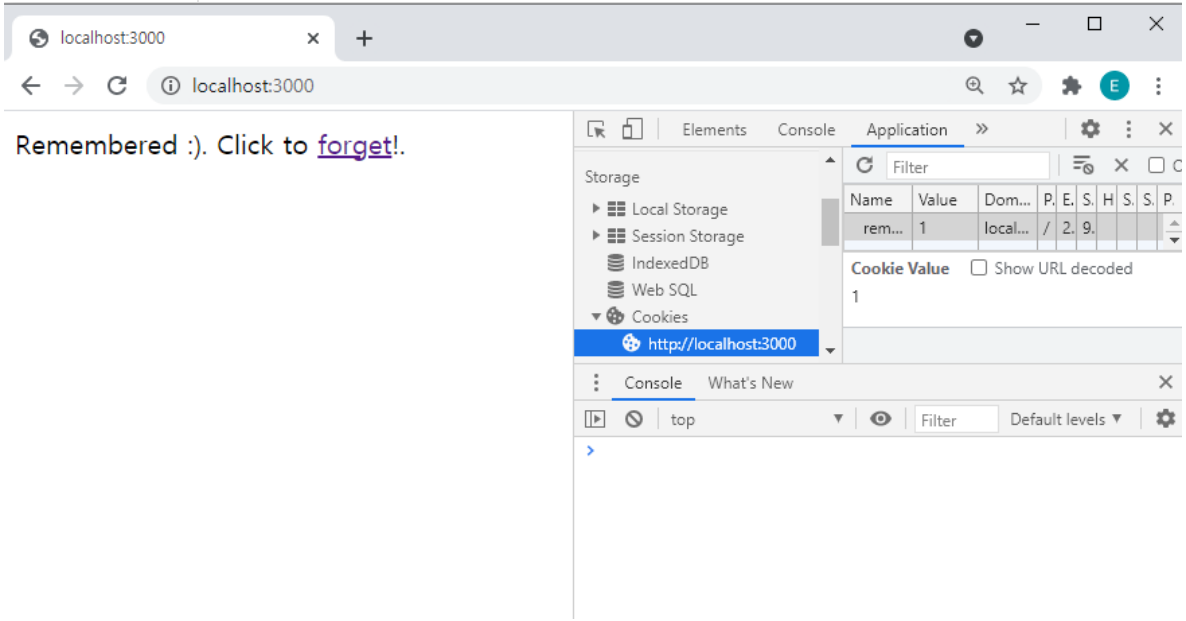
http://localhost:3000/images/1.jpg

Cookies Management

- You can send cookies to a Node.js server which can handle the same using the following middleware option.

```
var express      = require('express')
var cookieParser = require('cookie-parser')

var app = express()
app.use(cookieParser())

app.get('/', function(req, res) {
    console.log("Cookies: ", req.cookies)
    res.send('Hello World');
})
app.listen(3000)
```


localhost:3000
localhost:3000
Check to remember me Submit.
Elements
Console
Sources
Application
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
http://localhost:3000
Cache
Cache Storage
Cookies
Learn more
Console
What's New
top
Filter
Default levels


localhost:3000
localhost:3000
Remembered :). Click to forget!.
Elements
Console
Application
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
http://localhost:3000
Filter
Name
Value
Dom...
rem...
1
local...
Cookie Value
Show URL decoded
1
Console
What's New
top
Filter
Default levels

```
var express      = require('express')
var app = express()

var cookieParser = require('cookie-parser')
var logger = require('morgan');

app.use(logger(':method :status'))
app.use(cookieParser());

app.listen(3000, function() {
    console.log("App listening on port 3000...")
})

// parses x-www-form-urlencoded
app.use(express.urlencoded({ extended: false }))
```

```
app.get('/', function(req, res){
  if (req.cookies.remember) {
    res.send('Remembered :). Click to <a href="/forget">forget</a>!.');
  } else {
    res.send('<form method="post"><p>Check to <label>'
      + '<input type="checkbox" name="remember"/> remember me</label> '
      + '<input type="submit" value="Submit"/>.</p></form>');
  }
});

app.get('/forget', function(req, res){
  res.clearCookie('remember');
  res.redirect('back');
});

app.post('/', function(req, res){
  var minute = 60000;
  if (req.body.remember) res.cookie('remember', 1, { maxAge: minute });
  res.redirect('back');
});
```

Node.js

- **Node.js**
  - HTTP
  - FS
  - URL
  - NPM
  - Event
- **Express**
  - Routing methods
  - Middleware
  - Serving static files
  - format, multiparty, logger, cookie-parser