함수 선언문으로 함수 정의

```
function fact(n) {
    if (n <= 1) return n;
    return n*fact(n-1);
}
```

객체 생성법

리터럴 이용

```
var card2 = {suit : '하트', rank : 'A'};

console.log(card2.suit) // 하트 출력

card2.value = 14; // value 속성 추가
delete card2.rank; // rank 속성 삭제

console.log(card2);
console.log("rank" in card2); // false
console.log("suit" in card2); // true
```

new 연산자 사용

```
function Card(suit, rank) {
    this.suit = suit;
    this.rank = rank;
}

var card1 = new Card("하트", "A");
```

circle 객체

```
var circle = { center : { x : 1.0, y : 2.0 }, radius : 2.5, };

console.log('원의 중심 좌표는 (' + circle.center.x + "," + circle.center.y + ')');
console.log('원의 반지름은' + circle.radius);

const area = (radius) => (Math.PI * radius * radius);
const round = (radius) => (Math.PI * 2 * radius);
const translate = (x,y) => {
    circle.center.x += x;
    circle.center.y += y;
};

console.log("원의 면적은 " + area(circle.radius).toFixed(2) + "입니다");
console.log("원의 둘레는 " + round(circle.radius).toFixed(2) + "입니다");

translate(1,2);
console.log("(1,2) 이동한 원의 중심좌표는 (" + circle.center.x + "," +
circle.center.y + ")");
```

DOM을 사용한 이벤트 처리기 등록

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "utf-8">
        <title>시각을 콘솔에 표시하기</title>
        <script>
            function displayTime() {
                var d = new Date();
                console.log('현재 시각은 ' + d.toLocaleString() + " 입니다.");
            }

            window.onload = function() { // onload 프로퍼티에 함수 저장
                // input 요소 객체
                var button = document.getElementById("button");

                // 이벤트 처리기 등록
                button.onclick = displayTime;
            }
        </script>
    </head>
    <body>
        <input type = "button" value = "click" id = "button">
    </body>
</html>
```

innerHTML 사용

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset = "utf-8">
    <title>체질량지수 계산하기</title>
    <script>
        window.onload = function() {
            document.getElementById('calculate').onclick = function() {
                var weight = parseFloat(document.getElementById("weight").value)
                var height = parseFloat(document.getElementById("height").value)

                var bmi = document.getElementById('bmi');
                bmi.innerHTML = (weight / height / height).toFixed(1);
            }
        }
    </script>
    </head>
    <body>
        키 : <input type = "number" id = "height"> m <br>
        몸무게 : <input type = "number" id = "weight"> kg <br>
        당신의 체질량 지수는 <output id = "bmi"> </output> 입니다<br>
        <input type = "button" id = "calculate" value = "계산">
    </body>
</html>
```

타이머 함수 이용

```
<script>
    window.onload = function() {
```

```
        var time  = 0;
        var timer;

        document.getElementById('start').onclick = function() {
            timer = setInterval(function() {
                time += 0.01;
                var timeText = document.getElementById('sec');
                timeText.innerHTML = time.toFixed(2);
            }, 10);
        }

        document.getElementById('stop').onclick = function() {
            clearInterval(timer);
        };

        document.getElementById('reset').onclick = function() {
            time = 0;
            var timeText = document.getElementById('sec');
            timeText.innerHTML = time.toFixed(2);
            stopTime();
        }
    }
</script>
```

require, exports

```
// foo.js
exports.a = 10

// bar.js
const foo = require('./foo')
console.log(foo.a)
```

```
// bar.js
const foo = require('./foo')
const checknum = require('./fun')

console.log(foo.a);
checknum(10);

// foo.js
exports.a = 10;

// fun.js
function checknum(num) {
    if (num % 2) {
        console.log('홀수');
    } else {
       console.log('짝수');
    }
}

module.exports = check;
```

DOM 요소 접근 - 이벤트

```
<script>
    var big_pic = document.querySelector('#cup');
    var small_pics = document.querySelector('.small');

    for (var i = 0; i<small_pics.length;i++) {
        small_pics[i].onclick = showBig;
    }

    function showBig() {
        big_pic.setAttribute('src', this.src);
    }
</script>
```

HTTP 모듈

Server, listen

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

http.createServer((req, res) => {
    res.write('<h1>Hello Node!</h1>');
    res.end('<p>Hello Server!</p>');
}).listen(8080, () => {
    console.log('8080번 포트에서 대기중입니다');
})
```

파일 시스템 접근

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

fs.readFile('./readme.txt', (err, data) => {
    if (err) {
        throw err;
    }
    console.log(data);
    console.log(data.toString());
});
```

파일 만들기

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

fs.writeFile('./writeme.txt', '글이 입력됩니다', (err) => {
    if (err) {
        throw err;
    }
    fs.readFile('./writeme.txt', (err, data) => {
        if (err) {
           throw err;
        }
        console.log(data.toString());
    })
})
```

HTML 읽어서 전송하기

```
const http = require('http');
const fs = require('fs');

http.createServer((req, res) => {
    fs.readFile('./server2.html', (err, data) => {
        if (err) {
           throw err;
        }
        res.write(data);
        res.end();
    });
}).listen(8080, () => {
    console.log('서버 오픈');
});
```

onRequest Callback 처리

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

function onRequest(request, response) {
    var pathname = url.parse(request.url).pathname;
    console.log("Request for " + pathname + " received.");
    response.writeHead(200, {"Content-Type" : "text/plain"});
    response.write("Hello World");
    response.end();
}

http.createServer(onRequest).listen(8888);
console.log("Server has started");
```

Callback 실습

```
function findUser(id) {
    const user = {
        id : id,
        name : "User" + id,
        email : id + "@test.com"
    };
    return user;
}

function findUserAndCallBack(id, userFunc) {
  return userFunc(id);
}

const user = findUserAndCallBack(1, findUser);
console.log("user:", user);
```

URL

서로 다른 Path에 응답

```
const http = require('http');
const url = require('url');
const { URL } = url;
```

```
http.createServer((request, response) => {
  const parsedurl = url.parse(request.url)
  const resource = parsedurl.pathname;

  console.log('resource path=/',resource);

  response.writeHead(200, {'Content-Type':'text/plain; charset=utf-8'});
  if (resource == '/') response.end('안녕하세요');
  else if (resource == '/address') response.end('서울특별시 강남구 논현동 111');
  else if (resource == '/phone') response.end('전화번호');
  else if (resource == '/name') response.end('이름');
  else response.end('404 PAGE NOT FOUND');
}).listen(8080, () => {
  console.log('8080번 포트에서 서버 대기중입니다');
})
```

queryData에 응답

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

var app = http.createServer((req, res) => {
  var _url = req.url; // XXX:
  var queryData = url.parse(_url, true).query;

  // console.log(_url, queryData);

  // console.log(queryData);

  if (_url == "/") {
    _url = res.end('Welcome');
  }
  // else if (_url = "/favicon.ico") {
  //   return res.writeHead(404);
  // }

  res.writeHead(200);
  res.end(queryData.id);
});

app.listen(3000);
```

Query String을 이용한 동적 웹페이지

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

var app = http.createServer(function(request, response) {
  var _url = request.url;
  var queryData = url.parse(_url, true).query;
  var pathname = url.parse(_url, true).pathname;
  var title = queryData.id;
```

```
  if (pathname == '/') {
    fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description) {

      if (queryData.id == undefined) {
        title = 'Home';
        var description = 'Hello Node JS';
      }

      var template = `
        <!DOCTYPE html>
        <html>
          <head>
          <title>노드 웹서버 ${title}</title>
          <meta charset = "utf-8">
          </head>
          <body>
            <h1> <a href = "/">>WELCOME ${title}</a></h1>
            <ul>
              <li> <a href = "/?id=HTML">HTML</a></li>
              <li> <a href = "/?id=CSS">CSS</a></li>
              <li> <a href = "/?id=JavaScript">JavaScript</a></li>
            </ul>
            <h2>${title}</h2>
            <p>${description}</p>
          </body>
        </html>
      `;
    response.writeHead(200);
    response.end(template);
    })
  } else {
    response.writeHead(404);
    response.end('Not Found');
  }
});

app.listen(3000);
```

readdir 활용(디렉토리 리스트 생성, 표시)

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

function templateHTML(title, list, body){
  return `
  <!doctype html>
  <html>
  <head>
    <title>WEB1 - ${title}</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1><a href="/">WEB</a></h1>
    ${list}
    ${body}
  </body>
```

```
  </html>
  `;
}
function templateList(filelist){
  var list = '<ul>';
  var i = 0;
  while(i < filelist.length){
    list = list + `<li><a href="/?id=${filelist[i]}">${filelist[i]}</a></li>`;
    i = i + 1;
  }
  list = list+'</ul>';
  return list;
}

var app = http.createServer(function(request,response){
    var _url = request.url;
    var queryData = url.parse(_url, true).query;
    var pathname = url.parse(_url, true).pathname;
    if(pathname === '/'){
      if(queryData.id === undefined){
        fs.readdir('./data', function(error, filelist){
          var title = 'Home';
          var description = 'Hello, Node.js';
          var list = templateList(filelist);
          var template = templateHTML(title, list, `<h2>${title}
</h2>${description}`);
          response.writeHead(200);
          response.end(template);
        })
      } else {
        fs.readdir('./data', function(error, filelist){
          fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description)
{
            var title = queryData.id;
            var list = templateList(filelist);
            var template = templateHTML(title, list, `<h2>${title}
</h2>${description}`);
            response.writeHead(200);
            response.end(template);
          });
        });
      }
    } else {
      response.writeHead(404);
      response.end('Not found');
    }
});
app.listen(3000);
```

이벤트 처리

```
process.on('exit', function() {
    console.log('exit 이벤트 발생');
});
// process 객체 -> 내부적으로 EventEmitter 상속

setTimeout(function() {
    console.log('2초 후에 시스템 종료 시도함');
    process.exit();
}, 2000);
```

Buffer

```
const buffer = Buffer.from('저를 버퍼로 바꿔보세요');
console.log('from():', buffer);
console.log('length:', buffer.length);
console.log('toString():', buffer.toString());

const array = [Buffer.from('띄엄 '), Buffer.from('띄엄'), Buffer.from('띄어쓰
기')];
const buffer2 = Buffer.concat(array);
console.log('concat():', buffer2.toString());

const buffer3 = Buffer.alloc(5);
console.log('alloc():', buffer3);
```

createReadStream

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

const readStream = fs.createReadStream('./readme3.txt', {highWaterMark:16});
const data = [];

readStream.on('data', (chunk) => {
    data.push(chunk);
    console.log('data :', chunk, chunk.length);
});

readStream.on('end', () => {
    console.log('end :', Buffer.concat(data).toString());
});

readStream.on('error', (err) => {
    console.log('error :', err);
})
```

글 생성 UI 만들기(create, process_create, update 등)

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

function templateHTML(title, list, body){
  return `
  <!doctype html>
```

```
  <html>
  <head>
    <title>WEB1 - ${title}</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1><a href="/">WEB</a></h1>
    ${list}
    <a href="/create">create</a>
    ${body}
  </body>
  </html>
  `;
}
function templateList(filelist){
  var list = '<ul>';
  var i = 0;
  while(i < filelist.length){
    list = list + `<li><a href="/?id=${filelist[i]}">${filelist[i]}</a></li>`;
    i = i + 1;
  }
  list = list+'</ul>';
  return list;
}

var app = http.createServer(function(request,response){
    var _url = request.url;
    var queryData = url.parse(_url, true).query;
    var pathname = url.parse(_url, true).pathname;
    if(pathname === '/'){
      if(queryData.id === undefined){
        fs.readdir('./data', function(error, filelist){
          var title = 'Home';
          var description = 'Hello, Node.js';
          var list = templateList(filelist);
          var template = templateHTML(title, list, `<h2>${title}
</h2>${description}`);
          response.writeHead(200);
          response.end(template);
        })
      } else {
        fs.readdir('./data', function(error, filelist){
          fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description)
{
            var title = queryData.id;
            var list = templateList(filelist);
            var template = templateHTML(title, list, `<h2>${title}
</h2>${description}`);
            response.writeHead(200);
            response.end(template);
          });
        });
      }
    } else if (pathname == '/create') {
        fs.readdir('./data', function(error, filelist) {
            var title = 'WEB - create';
            var list = templateList(filelist);
            var template = templateHTML(title, list, `
```

```
                <form action = "http://localhost:3000/process_create" method =
"post">
                <p> <input type = "text" name = "title" placeholder = "title">
</p>
                <p>
                <textarea name = "description" placeholder = "description">
</textarea>
                </p>
                <p>
                <input type = "submit">
                </p>
                `);
            response.writeHead(200);
            response.end(template);
        });
    } else if (pathname == '/process_create') {
        var body = '';
        request.on('data', function(data) {
            body = body + data;
        })

        request.on('end', function() {
            var post = qs.parse(body);
            var title = post.title;
            var description = post.description

            fs.writeFile(`data/${title}`, description, 'utf8', function(err) {
                response.writeHead(302, {Location : `/?id=${title}`});
                response.end();
            });
      });
    } else {
      response.writeHead(404);
      response.end('Not found');
    }
});
app.listen(3000);
```