Date: / /

Note:

Mon Tue Wed Thu Fri Sat

Chapter 3
Socket Programming

Intro to JavaScript

- JavaScript มีรูปแบบคล้ายภาษา C แต่ไม่เหมือน.
- "case sensitive" ตัวเล็ก / ตัวใหญ่ ไม่เหมือนกัน
- เวลาจบคำสั่ง จะมี ; หรือไม่มีก็ได้.
- "Block" ใช้เครื่องหมาย {...}
  ถ้าประกาศตัวแปรไว้ใน block จะใช้นอก blockได้
- การประกาศตัวแปรใช้ var
  ถ้าลำตัวแปรมาใช้โดยยังไม่ประกาศ โดยจะเป็นตัวแปร "local variable"

ชนิดตัวแปร

- Boolean ชนิด ไม่ก็จริง.
- Number ตัวเลขทั่วไป double
- String
- Special values. "null and undefined" ไม่ได้ตัวระบุค่า

Objects เก็บข้อมูลที่เป็นชื่อของ value.

name : value

เช่น objBob = { name: "Bob", grade : 'A'; level : 3}.

ถ้าต้องการเพิ่มข้อมูล ชื่อการ์ตูน objBob.fullname = 'Robert';

Function

- Anonymous function
  function ที่ไม่ต้องกำหนดชื่อ function ในการใช้งาน เราจะต้องสร้างตัวแปรขึ้นมาเก็บตัวแล้วนำไปต้องอ้างกับ function

ex1: var square = function (x)
{ return x*x; }.

Date: / /

Mon Tue Wed Thu Fri Sat Sun

Note:

ex2: var hello = function ()
{ alert ("Hello"); }

การเรียก function ต้องเรียกผ่านตัวแปรที่อ้างอิงได้กับ function พร้อมกำหนด Argument (ถ้ามี)

เช่น ex1: var sq = square (10) // sq = 100

ex2: hello ();

▶ Curried function

คือการทำให้ function ที่รับหลายๆ Parameters แยกออกเป็น function ที่รับแค่ Parameter เดียว หลายๆ function ซ้อนกันไป.

เช่น function Curried Add (x)
{ return function (y)
{ return x+y; }
};

g = CurriedAdd (2); → 2: แทนค่าใน x

g (3) → 3: แทนค่าใน y.

♥ Intro to Node.js

- หลายๆ คนดีได้ง่าย.
- กินกำลังไม่มาก ไม่ต้องออกแรง
- ทำงานผ่าน JavaScript's callback functionality. ซึ่งเป็นคุณสมบัติของ non-blocking I/O
- Event-loops.
- Non-blocking I/O
- Node.js คือ JavaScript ในการเขียน แต่เป็น JavaScript ตามสิ่งที่ run บน browser
- ทุกอย่างของ Node.js run อยู่บน Single thread

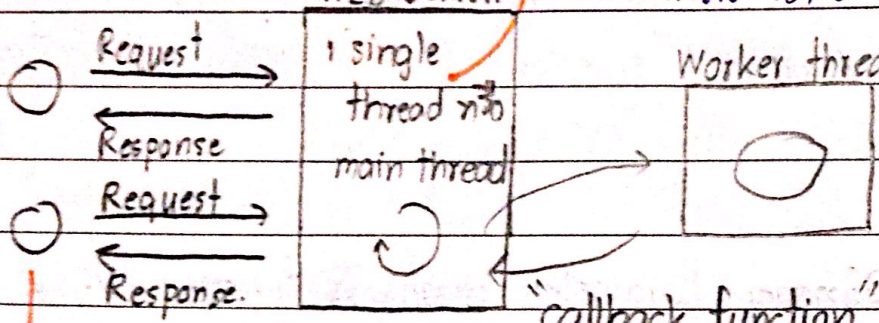
จะแยกการทำงานเป็นหลายๆ thread ซึ่งแต่ละ thread ก็จะต้องรอๆ กัน เมื่อทำงานเสร็จจะแจ้งผ่าน "callback function" ให้รอรับ function ให้กลับมาทำงาน.

Date: / /

Note:

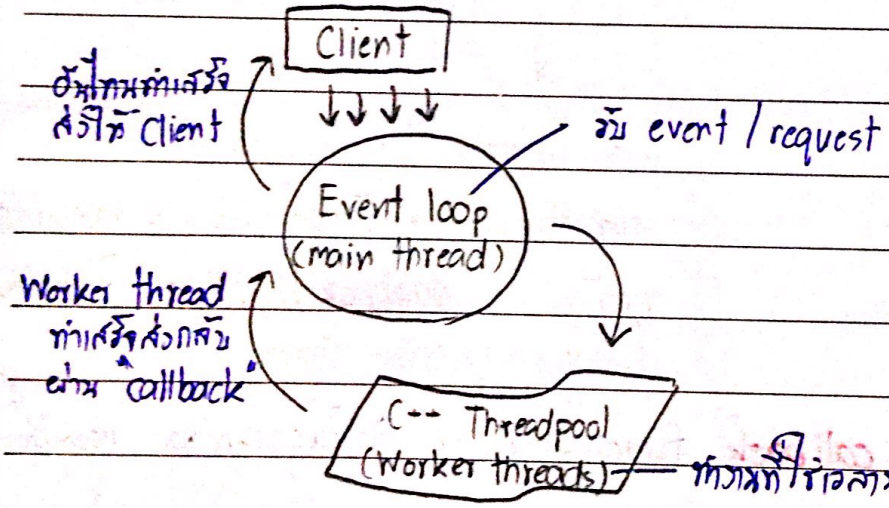
Single thread:

1 single thread / main thread คอยรับ event หรือ request แล้วส่งต่อการทำงานเสร็จใน main thread ก็ส่งต่อไปเลย แต่หากงานไหนใช้เวลานาน ก็จะส่งให้ "Worker thread" ทำต่อ ซึ่งเมื่อ Worker thread ทำเสร็จจะถอยกลับไปที่ main ผ่าน "callback function"



เริ่มแรกเรียกว่า 1 request ต่อ 1 process แต่จะกิน CPU กิน RAM เลยเปลี่ยนมาเป็น 1 request ต่อ 1 thread ซึ่งการแตก thread เยอะๆ ก็ให้ทำงานพร้อมกันได้ แต่ก็เปลี่ยนอีก เป็น ทุก request คือ 1 single thread.

Event-loops



Non-Blocking I/O

- Traditional I/O เช่น Blocking I/O จะโดน Block ไว้ก่อนจนกว่าข้างบนจะทำงานเสร็จ

```
ex. var result = db.query("select x from table_Y");
    doSomethingWith(result);      // wait for result
    doSomethingWithOutResult();   // execution is blocked
```

- Non-traditional, Non-blocking I/O จะไม่รอการทำงานด้านบน ทำงานเลย เพื่อไม่ให้เสียเวลา จะทำงานเร็วกว่า Traditional I/O.

Date: / /

Note:

Mon Tue Wed Thu Fri Sat Sun

ex. db.query (' select x from table_Y" function (result)
{ doSomethingWith (result); });
doSomethingWithOutResult();

→ call back function

Node.js Ecosystem

ex. var modulex = require ('./module');

→ ชื่อ module ต้องมี path หรือใส่ชื่อ module

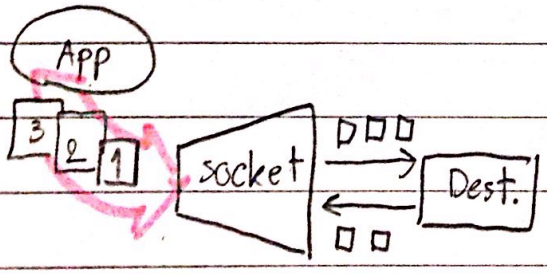
♥ Socket Programming

"Socket" คือ การเชื่อมต่อการสื่อสารระหว่างจุดสองจุด (เครื่อง 2 เครื่อง) แบบไปกลับ ตัวอย่าง 2 โปรแกรม (server กับ client) ตามในเครือข่ายเดียวกัน.

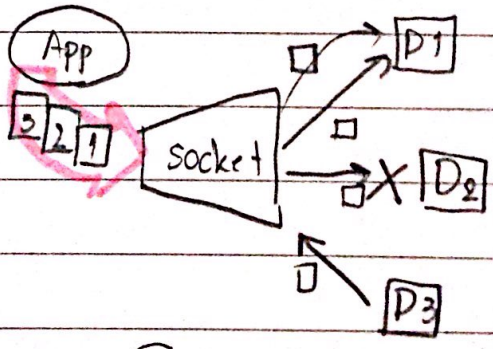
types of sockets.

▶ TCP Socket
- Stream-oriented
- reliable delivery
- in-order guaranted
- connection-oriented
- bidirectional

▶ UDP Socket
- Datagram-oriented.
- unreliable delivery.
- no order guarantees.
- no notion of "connection"
- can send or receive.



TCP : ส่งข้อมูลต้องถูกต้อง น่าเชื่อถือ แต่ช้ากว่า เพราะต้องรอการตอบกลับ



UDP: ผู้รับส่ง Datagram ทางเดียวไม่น่าเชื่อถือ อาจทาง ส่งข้อมูลไม่ถูกต้อง. แต่ส่งเร็ว.