Reading Review: Okamoto & Ohta's Ideal Digital Currency Properties

1. Independence
   - not dependent on any physical condition
   - can be transferred through computer networks
2. Security
   - cannot be copied and reused (double-spending)
   - cannot be forged
3. Privacy
4. Off-line Payment
   - No need to be linked to a host to process payment
5. Transferability
6. Divisibility

CS 168: Blockchain and Cryptocurrencies

JavaScript Crash Course

Prof. Tom Austin
San José State University

History of JavaScript

Brendan Eich

1995: Netscape hired
Brendan Eich.

His job: implement Scheme for the web browser.

After a few meetings,
Scheme was deemed too weird…

In 10 days, Brendan Eich wrote the initial version of JavaScript for Netscape 2.0 Beta.

Why JavaScript?

- Familiar syntax
  - *Superficially* similar to Java
- Rich set of libraries
  - Especially in the cryptocurrency world
- Concurrency model
  - Asynchronous model
  - Less prone to bugs
- Because I like it

Node.js

- Server-side JavaScript
- Based on Google's V8 engine
- npm: Node.js package manager
- http://nodejs.org/

JS: the good, the bad, and the ugly

- JS has good features – use these
- JS has other features
  - Take CS 152 w/ me if you want to know more

Avoid these features (though you may see them)

- `var` – use `let` instead
- Functions as constructors
- `==`
  - Use `===` instead
- `with`
- `eval`

Use these features

- `"use strict";`
- JSON data
- Classes
- Callback functions
- Arrow functions

Sample JS program

```
"use strict";
function addList(list) {
  let sum=0;
  for (let i=0; i<list.length; i++)
    sum += list[i];
  return sum;
}
console.log(addList([1,2,3,4]));
```

Warning: console.log is not standard (though widely supported)

JavaScript Object Notation (JSON)


```
{ "asset_id_base": "USD",
  "rates": [
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "LTC",
      "rate": 0.030537365914358224607146457
    },
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "BTC",
      "rate": 0.00028079567733887072036216 01
    }]
}
```

JSON keys


```
{ "asset_id_base": "USD",
  "rates": [
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "LTC",
      "rate": 0.030537365914358224607146457
    },
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "BTC",
      "rate": 0.000280795677338870720362160 1
    }]
}
```

JSON array


```
{ "asset_id_base": "USD",
  "rates": [
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "LTC",
      "rate": 0.030537365914358224607146457
    },
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "BTC",
      "rate": 0.000280795677338870720362160 1
    }]
}
```

obj['rates'][0]


```
{ "asset_id_base": "USD",
  "rates": [
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "LTC",
      "rate": 0.030537365914358224607146457
    },
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "BTC",
      "rate": 0.000280795677338870720362160 1
    }]
}
```

obj['rates'][1]


```
{ "asset_id_base": "USD",
  "rates": [
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "LTC",
      "rate": 0.030537365914358224607146457
    },
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "BTC",
      "rate": 0.00028079567733887072036216 01
    }]
}
```

```
obj['rates'][1]['asset_id_quote']
```


```
{ "asset_id_base": "USD",
  "rates": [
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "LTC",
      "rate": 0.030537365914358224607146457
    },
    { "time": "2019-01-25T23:47:01.6754729Z",
      "asset_id_quote": "BTC",
      "rate": 0.000280795677338870720362160 1
    }]
}
```

Lab: Cryptocurrency Price Converter (part 1 – parse JSON from file)

1. Download
   - `currencyConverter.js`
   - `rates.json` (contains cryptocurrency pricing)
2. Implement `readJsonFromFile`
   - Use `fs.readFileSync` to get file contents
   - Use `JSON.parse` to convert data to an object

See Canvas/course website for additional tips

ES6 Classes

- Largely similar to Java
- No typing information
- Syntactic sugar for *prototypes*
- Unavailable in older versions of JavaScript

JS class example

```
"use strict";

class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    console.log("...");
  }
}
```

```
class Cat extends Animal {
  constructor(name, favoriteFood) {
    super(name);
    this.favoriteFood = favoriteFood;
  }
  speak() {
    console.log('Meow');
  }
  eat(food) {
    if (this.favoriteFood === food) {
      console.log("Purr...");
    }
    console.log("munch, munch, munch");
  }
}
```

```
let garfield = new Cat("Garfield", "lasagna");

garfield.speak();
console.log(garfield.favoriteFood);

garfield.eat("chicken");
garfield.eat("lasagna");

class Fish extends Animal {
  constructor(name) {
    super(name);
  }
}

let frankie = new Fish("Frankie");
frankie.speak();
```

Lab: Cryptocurrency Price Converter (part 2 – calculate exchange rates)

1. Update `calculateRates` to add conversion rates between different cryptocurrencies.
   - Use USD values of both currencies to calculate exchange rate.
2. Uncomment the calls to the 'test' function toward the end of the file.

Expected output:

4000 ETH is worth 129.23225502950967 BTC.

200 BTC is worth 293589.0343862397 EOS.

JavaScript Concurrency

- JavaScript is single-threaded
- Concurrency is based on asynchronous model
  - Current process runs to completion
  - Other tasks wait their turn
- Avoid (many) *race conditions*
  - Race condition: results depend on order of execution
- Watch out for deadlocks
  - Deadlock: 2 processes waiting for each other to finish
- Note: JS does NOT support parallelism

JS Event model

- May be *synchronous* (default) or *asynchronous*
  - Synchronous: caller waits for result
  - Asynchronous: caller continues w/o waiting
- `EventEmitter` class methods
  - `emitter.emit` triggers an event
  - `emmitter.on` registers an *event handler*
- An event handler is a function called whenever an event is triggered

Note that JavaScript is single-threaded.

An event runs to completion before the next event begins.

```
"use strict";
let EventEmitter = require('events');
let ee = new EventEmitter();
let dead = false;
ee.on('die', function() {
  console.log("I'm melting!!!");
  dead = true;
});
setTimeout(function() {
  ee.emit('die');
}, 100);
while (!dead) {}
console.log('done');
```

Arrow functions

- Introduced in ES6
- Concise function syntax
- `this` bound lexically
  - Normal functions bind `this` dynamically

```
function sort (lst, fn) {
  for (let i=0; i<lst.length; i++) {
    for (let j=0; j<lst.length-1; j++) {
      if (fn(lst[i], lst[j])) {
        let tmp = lst[i];
        lst[i] = lst[j];
        lst[j] = tmp;
      }
    }
  }
}

let arr = [1,2,99,10,42,7,-3,88,6];
sort(arr, function(x,y) { return x<y; });
```

```
function sort (lst, fn) {
  for (let i=0; i<lst.length; i++) {
    for (let j=0; j<lst.length-1; j++) {
      if (fn(lst[i], lst[j])) {
        let tmp = lst[i];
        lst[i] = lst[j];
        lst[j] = tmp;
      }
    }
  }
}

let arr = [1,2,99,10,42,7,-3,88,6];
sort(arr, (x,y) => x<y);
```

Lab: Cryptocurrency Price Converter (part 3 – use event handlers)

Update CurrencyConverter to respond to events

- "SHOW_PRICE" event
  - Already registered in constructor
  - Print out the exchange rate between 2 currencies
- "UPDATE_USD_PRICE" event
  1. Update USD price for the specified cryptocurrency
  2. Update the exchange rates for this cryptocurrency with all other cryptocurrencies.

Sample output and more details available in Canvas and course website.

JavaScript references

- David Flanagan's "JavaScript: The Definitive Guide"
- Douglas Crockford's "JavaScript: The Good Parts"
- http://w3schools.com/js/default.asp
  - Intro to basics
- http://eloquentjavascript.net/
  - Free ebook on JS
- https://medium.com/technoetics/node-js-event-emitter-explained-d4f7fd141a1a
  - Review of EventEmitter's details
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes
  - Discussion about ES6 classes

Reading: Review before next class

- "Bitcoin and Cryptocurrency Technologies" (prepub version),
  - Sections 1.1, 1.3, and 1.4
  - lecture 1: https://youtu.be/fOMVZXLjKYo
- "Mastering Bitcoin", 2nd ed.
  - P. 55-64
- Mark Stamp's textbook
  - Chapters 2-5