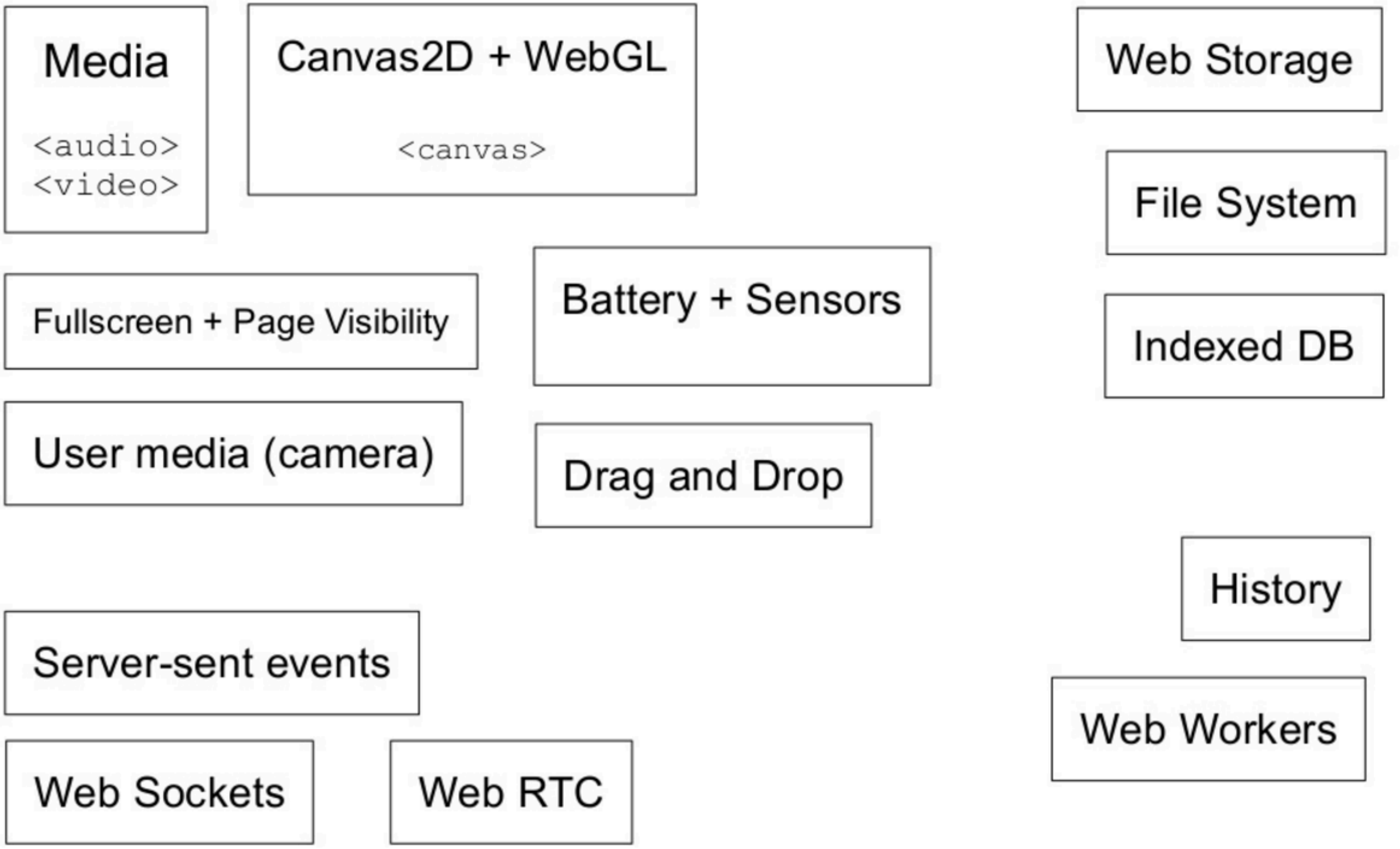
Lecture 08.
Client Side JavaScript Toolkits

Modern Web Programming

(http://my.ss.sysu.edu.cn/wiki/display/WEB/ supported by Deep Focus)

School of Data and Computer Science, Sun Yat-sen University

HTML5 JavaScript API



Media

`<audio>`
`<video>`

Canvas2D + WebGL

`<canvas>`

Web Storage

File System

Fullscreen + Page Visibility

Battery + Sensors

Indexed DB

User media (camera)

Drag and Drop

History

Server-sent events

Web Workers

Web Sockets

Web RTC

Marionette
js

BACKBONE.JS

BOWER

YO

ember

jQuery
write less, do more.

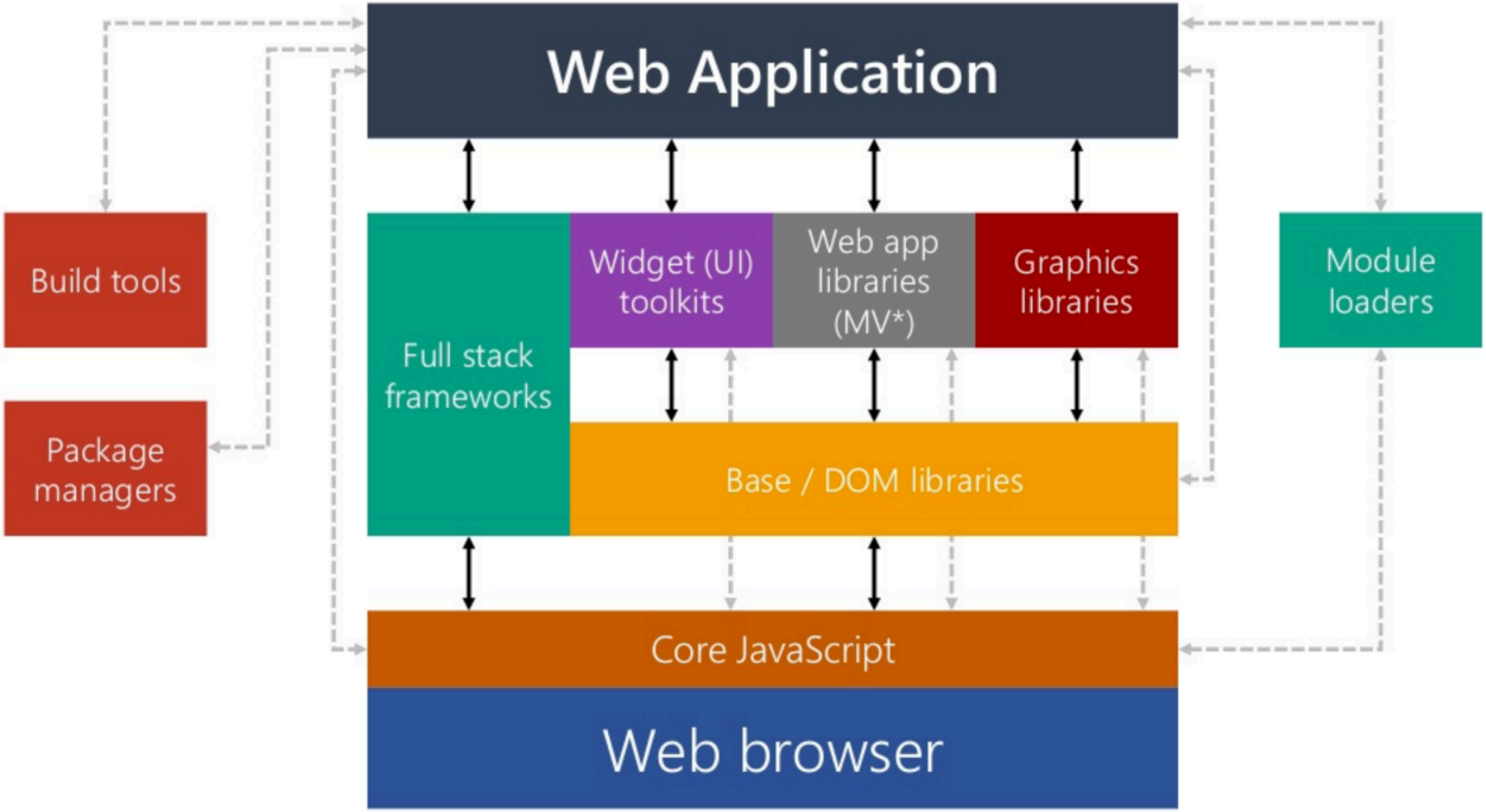
dōjō
toolkit

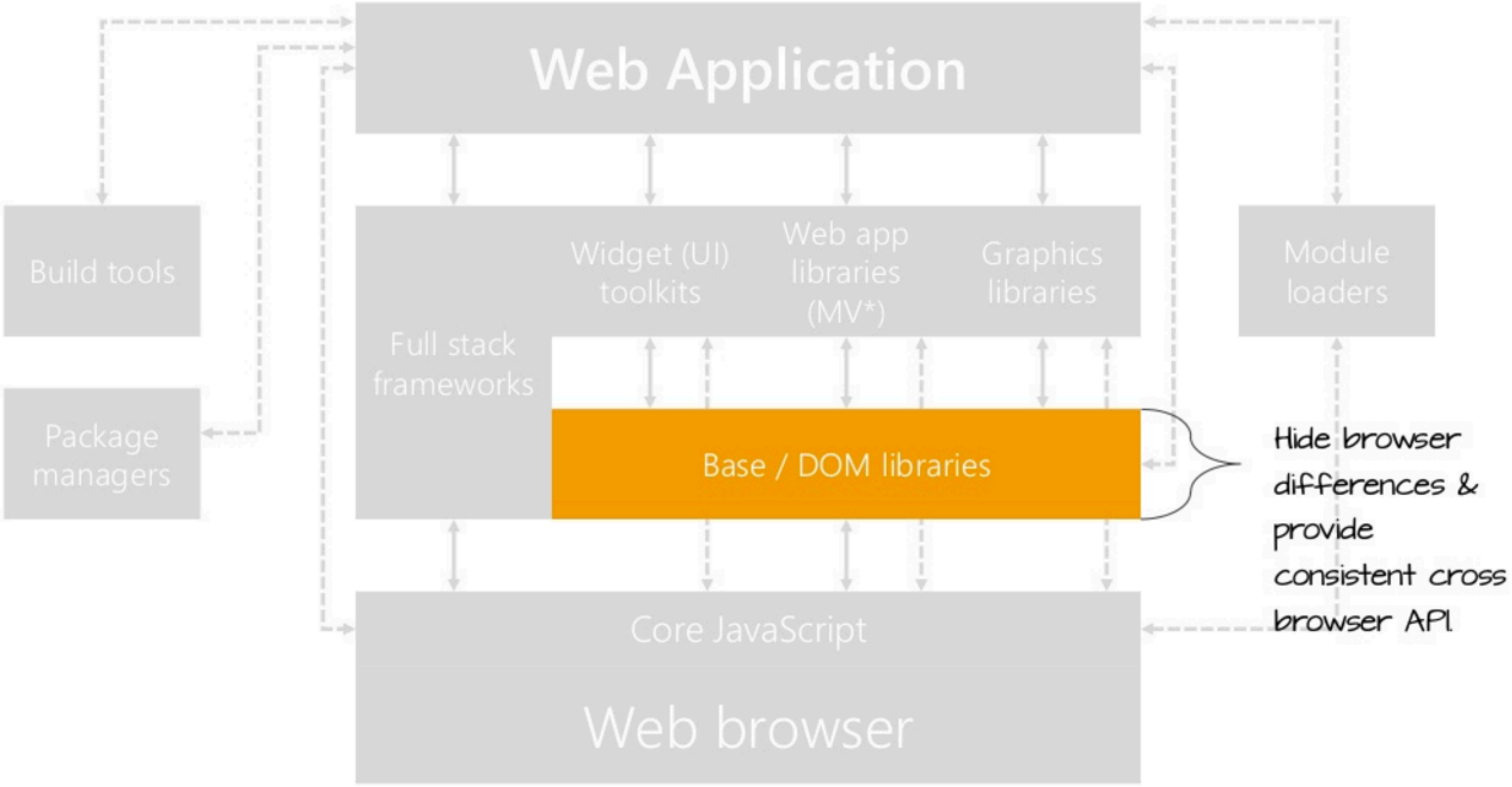
GRUNT

ANGULARJS
by Google

mootools

Cappuccino


Web Application
Build tools
Package managers
Full stack frameworks
Widget (UI) toolkits
Web app libraries (MV*)
Graphics libraries
Module loaders
Base / DOM libraries
Core JavaScript
Web browser



Hide browser differences & provide consistent cross browser API

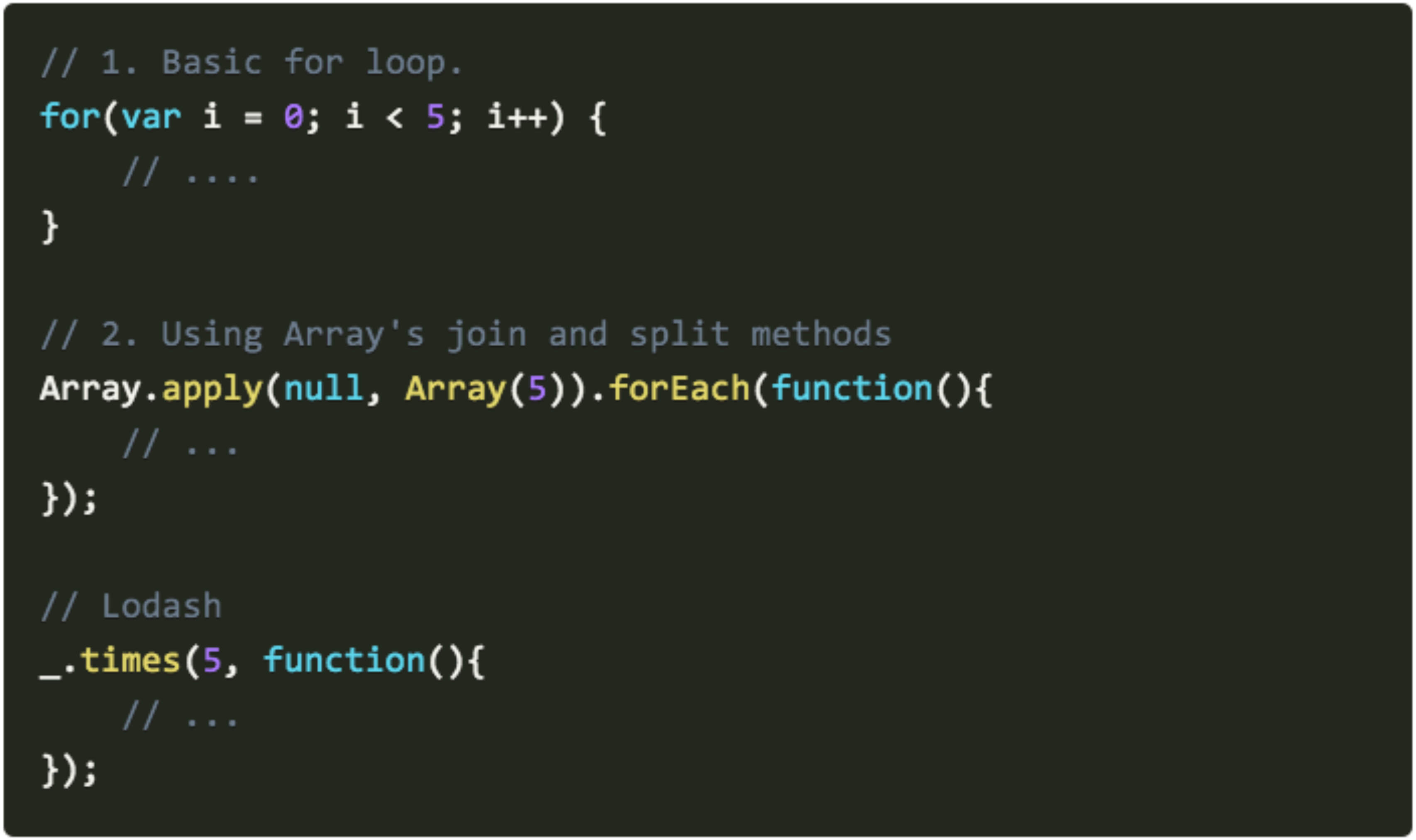
Part I jQuery

Part II lodash + moment

Colin Toh @p0larboy

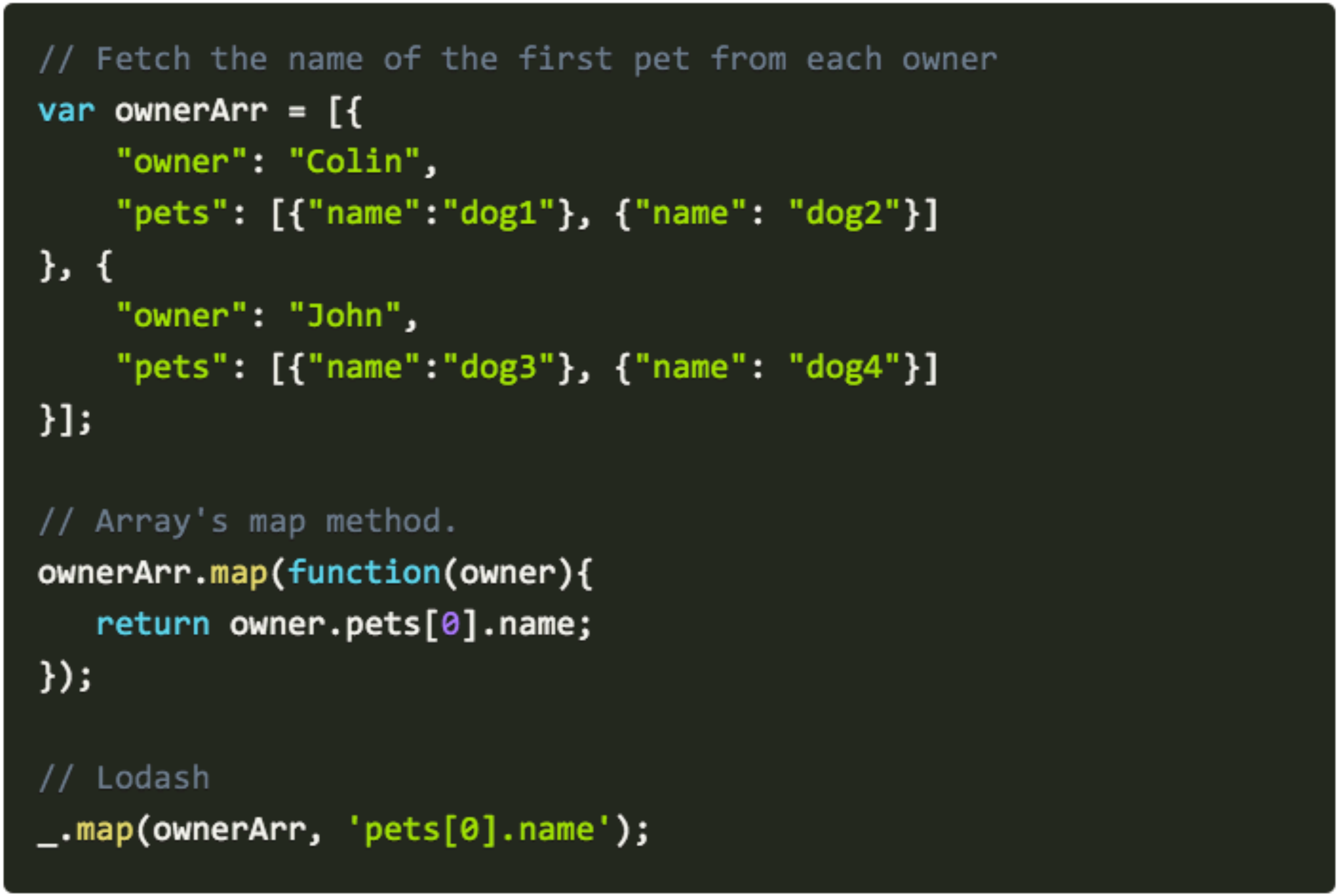
http://colintoh.com/blog/lodash-10-javascript-utility-functions-stop-rewriting

1) Loop for N times



```
// 1. Basic for loop.
for(var i = 0; i < 5; i++) {
    // ....
}

// 2. Using Array's join and split methods
Array.apply(null, Array(5)).forEach(function(){
    // ...
});

// Lodash
_.times(5, function(){
    // ...
});
```

2) Loop through a collection and return a deeply-nested property from each item



```
// Fetch the name of the first pet from each owner
var ownerArr = [{
    "owner": "Colin",
    "pets": [{"name":"dog1"}, {"name": "dog2"}]
}, {
    "owner": "John",
    "pets": [{"name":"dog3"}, {"name": "dog4"}]
}];

// Array's map method.
ownerArr.map(function(owner){
   return owner.pets[0].name;
});

// Lodash
_.map(ownerArr, 'pets[0].name');
```

3) Create an array of N size and populate them with unique values of the same prefix

```
// Create an array of length 6 and populate them with unique values. The value must be prefix with "ball_".
// eg. [ball_0, ball_1, ball_2, ball_3, ball_4, ball_5]

// Array's map method.
Array.apply(null, Array(6)).map(function(item, index){
    return "ball_" + index;
});



// Lodash
_.times(6, _.uniqueId.bind(null, 'ball_'));
```

Get rid of the .bind(null,...)

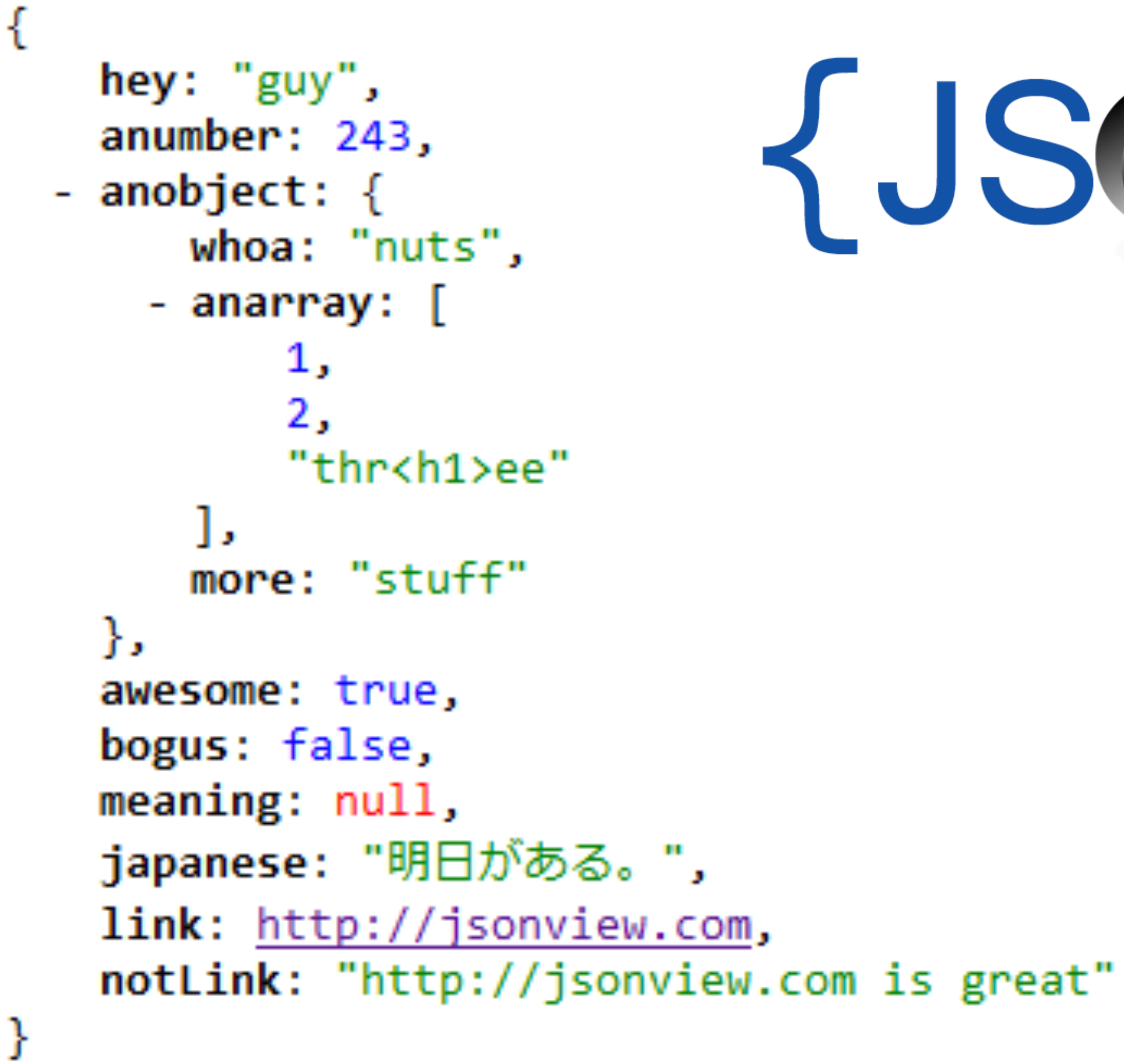
```
// Lodash
_.times(6, _.partial(_.uniqueId, 'ball_'));
```

4) Deep-cloning Javascript object

```
var objA = {
    "name": "colin"
}

// Normal method? Too long. See Stackoverflow for solution: http://stackoverflow.com/questions/4459928/how-to-deep-clone-in-javascript

// Lodash
var objB = _.cloneDeep(objA);
objB === objA // false
```

Deep-cloning javascript object is difficult and there is no easy way around it. Altenative naive solution: `JSON.parse(JSON.stringify(objectToClone))` for deep-cloning. However, this will only work if there are no function within the object.

{JSON}



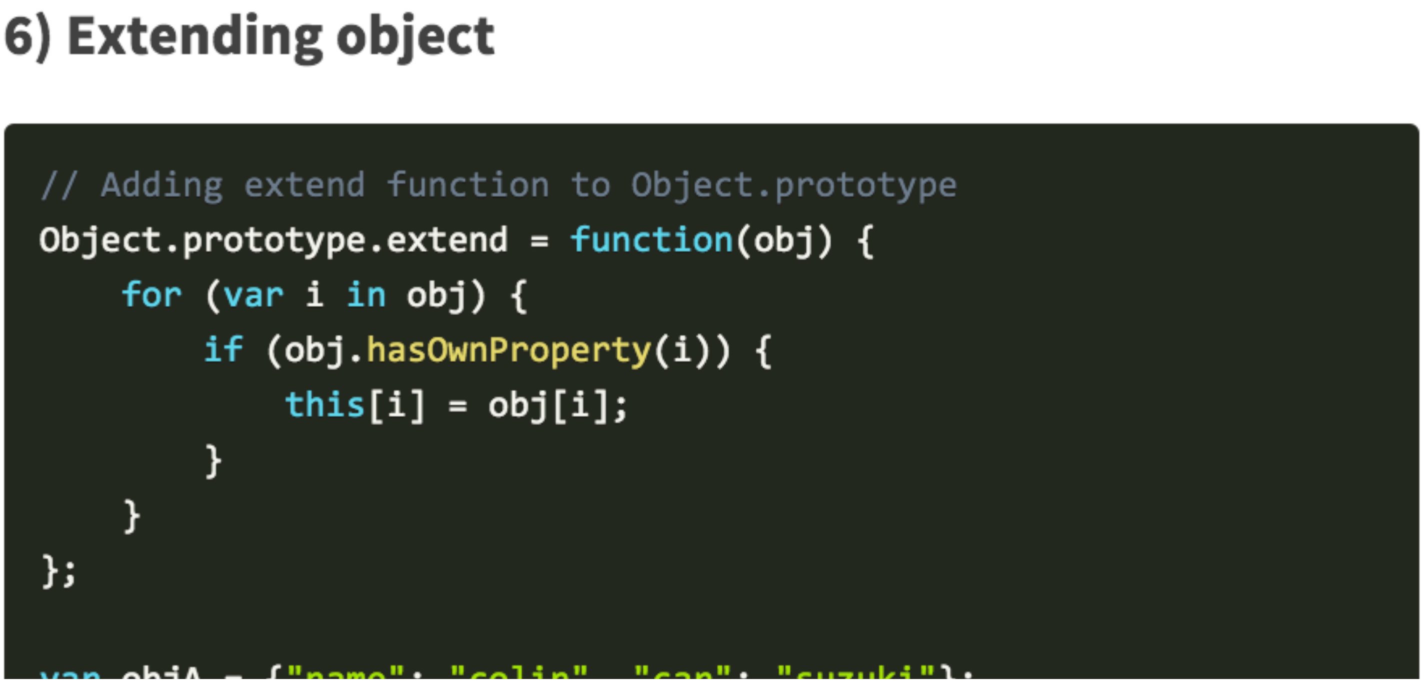
```
{
    hey: "guy",
    anumber: 243,
  - anobject: {
        whoa: "nuts",
      - anarray: [
            1,
            2,
            "thr<h1>ee"
        ],
        more: "stuff"
    },
    awesome: true,
    bogus: false,
    meaning: null,
    japanese: "明日がある。",
    link: http://jsonview.com,
    notLink: "http://jsonview.com is great"
}
```

5) Get Random Number between a range

```
// Get a random number between 15 and 20.

// Naive utility method
function getRandomNumber(min, max){
    return Math.floor(Math.random() * (max - min)) + min;
}

getRandomNumber(15, 20);

// Lodash
_.random(15, 20);
```

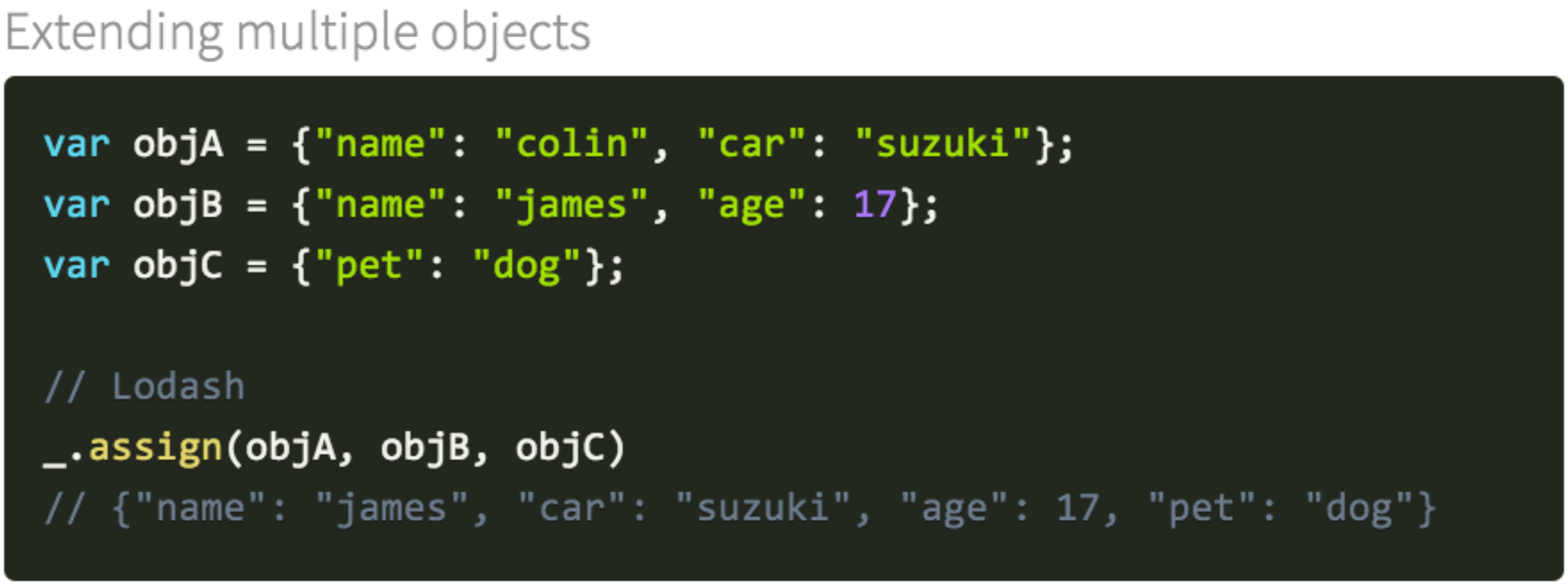
Additional option for _.random

```
_.random(20); // Return random number between 0 to 20
_.random(15, 20, true); // Return random floating numbers between
 15 and 20
```



6) Extending object

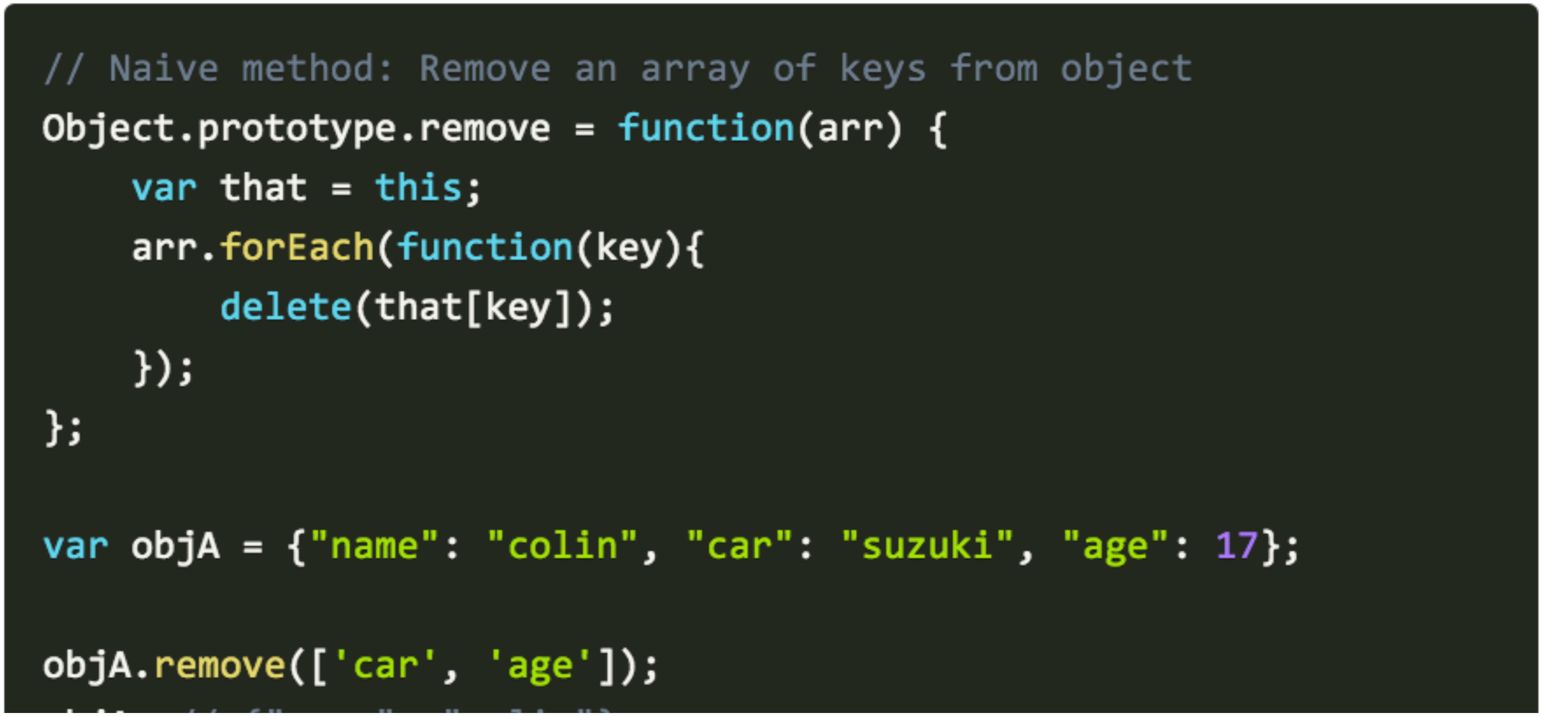
```
// Adding extend function to Object.prototype
Object.prototype.extend = function(obj) {
    for (var i in obj) {
        if (obj.hasOwnProperty(i)) {
            this[i] = obj[i];
        }
    }
};

var objA = {"name": "colin", "car": "suzuki"};
```



Extending multiple objects

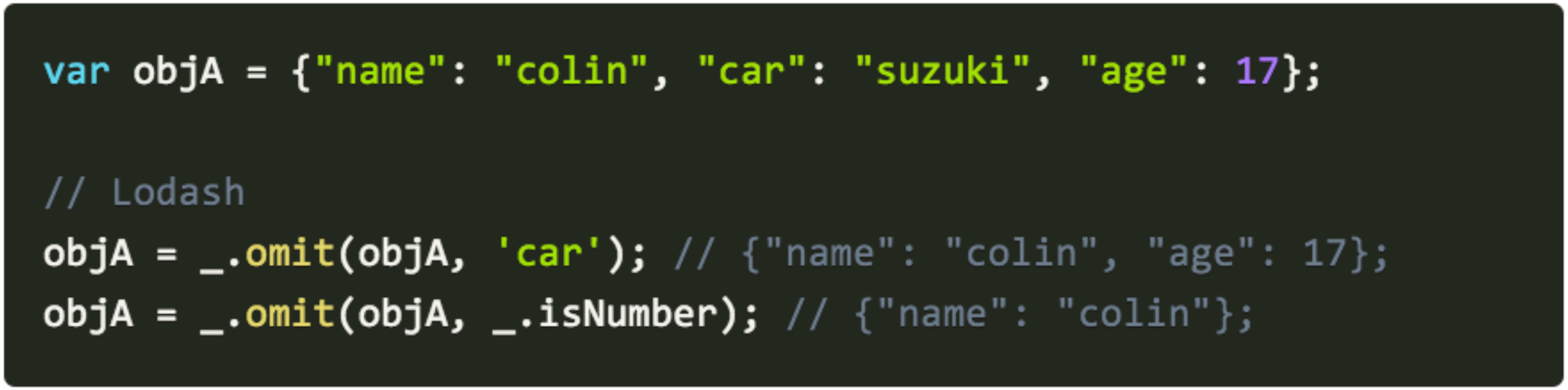
```
var objA = {"name": "colin", "car": "suzuki"};
var objB = {"name": "james", "age": 17};
var objC = {"pet": "dog"};

// Lodash
_.assign(objA, objB, objC)
// {"name": "james", "car": "suzuki", "age": 17, "pet": "dog"}
```

7) Removing properties from object



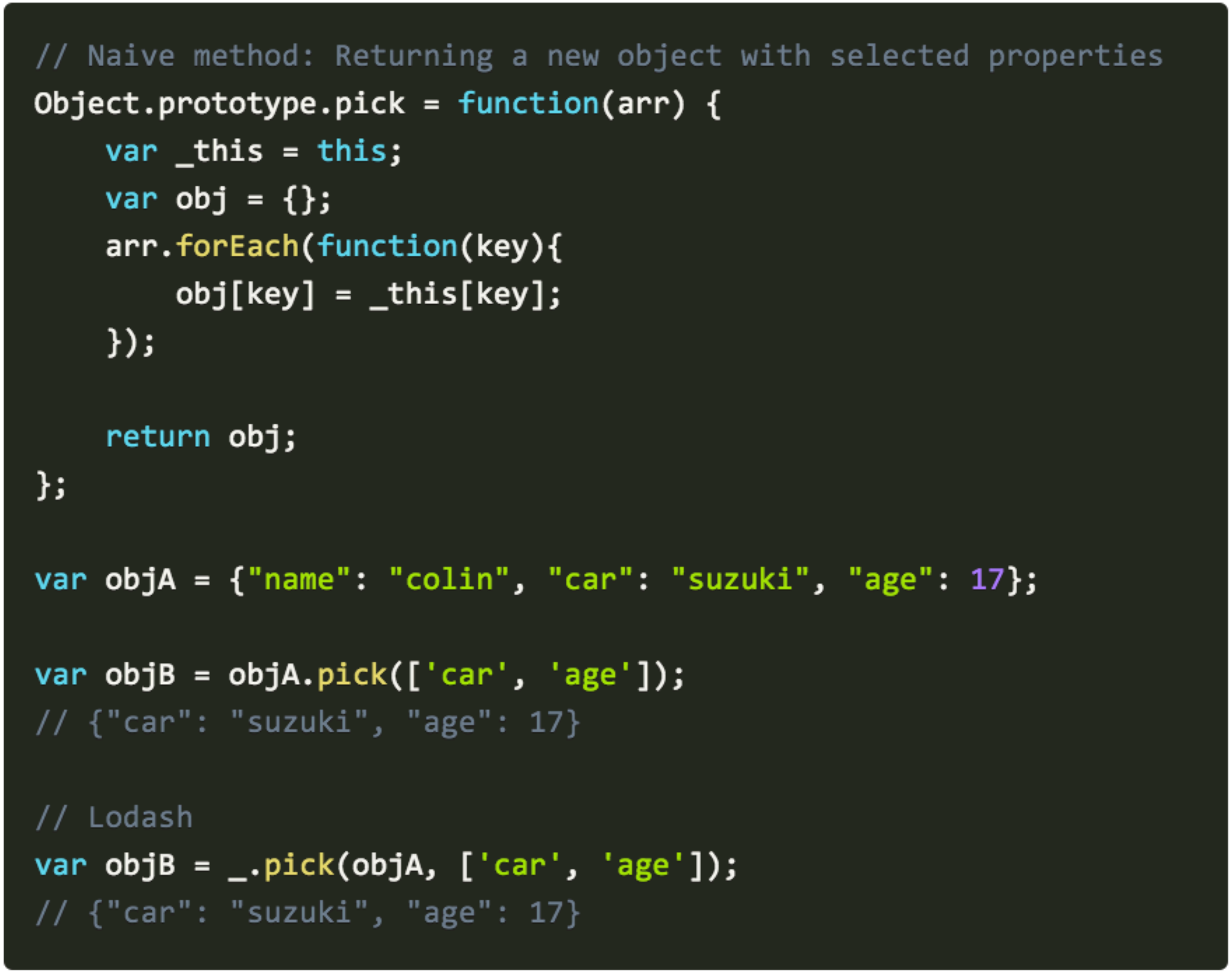
```
// Naive method: Remove an array of keys from object
Object.prototype.remove = function(arr) {
    var that = this;
    arr.forEach(function(key){
        delete(that[key]);
    });
};

var objA = {"name": "colin", "car": "suzuki", "age": 17};

objA.remove(['car', 'age']);
```

More use-cases

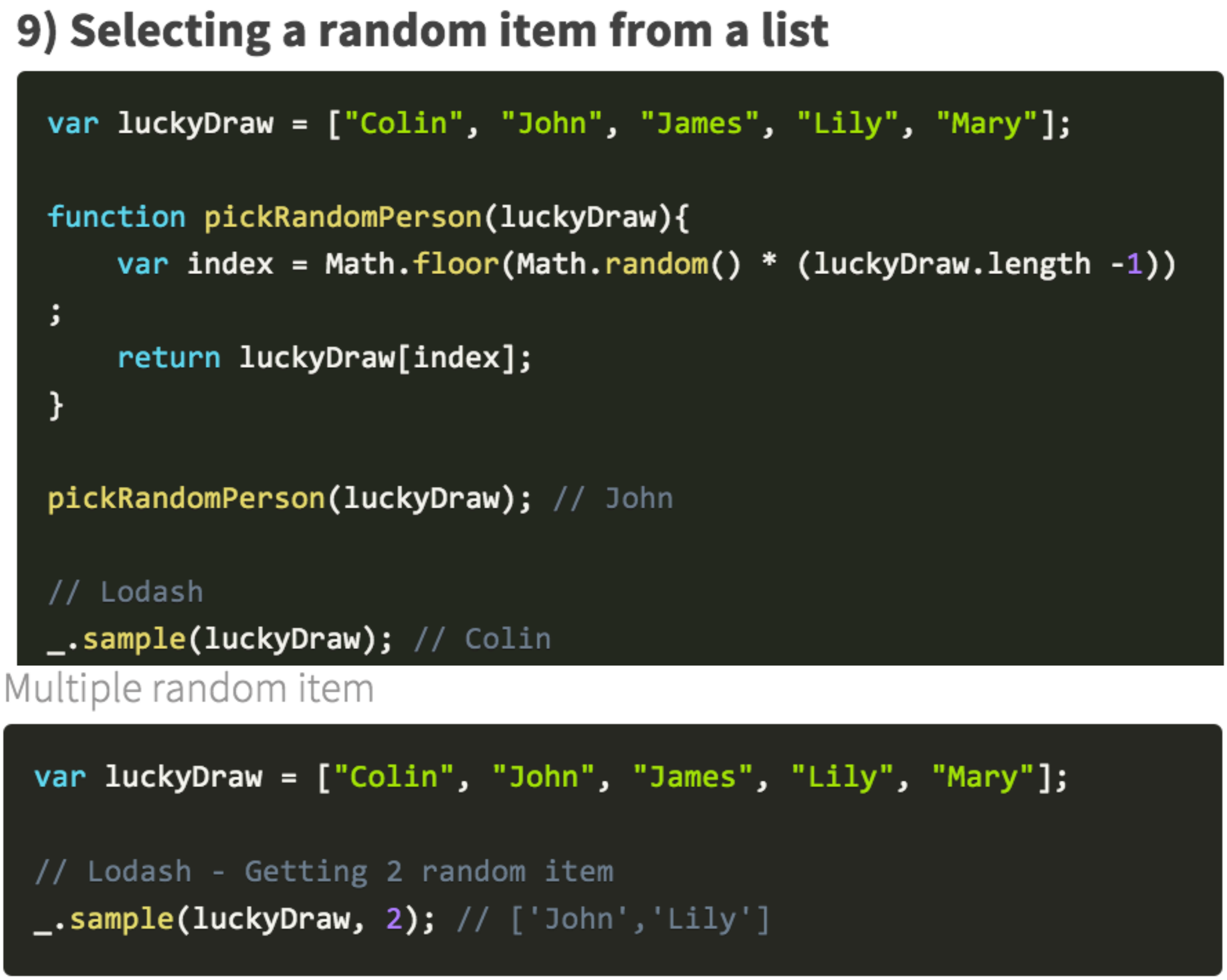


```
var objA = {"name": "colin", "car": "suzuki", "age": 17};

// Lodash
objA = _.omit(objA, 'car'); // {"name": "colin", "age": 17};
objA = _.omit(objA, _.isNumber); // {"name": "colin"};
```

8) Select properties from another object to form new object



```
// Naive method: Returning a new object with selected properties
Object.prototype.pick = function(arr) {
    var _this = this;
    var obj = {};
    arr.forEach(function(key){
        obj[key] = _this[key];
    });

    return obj;
};

var objA = {"name": "colin", "car": "suzuki", "age": 17};

var objB = objA.pick(['car', 'age']);
// {"car": "suzuki", "age": 17}

// Lodash
var objB = _.pick(objA, ['car', 'age']);
// {"car": "suzuki", "age": 17}
```



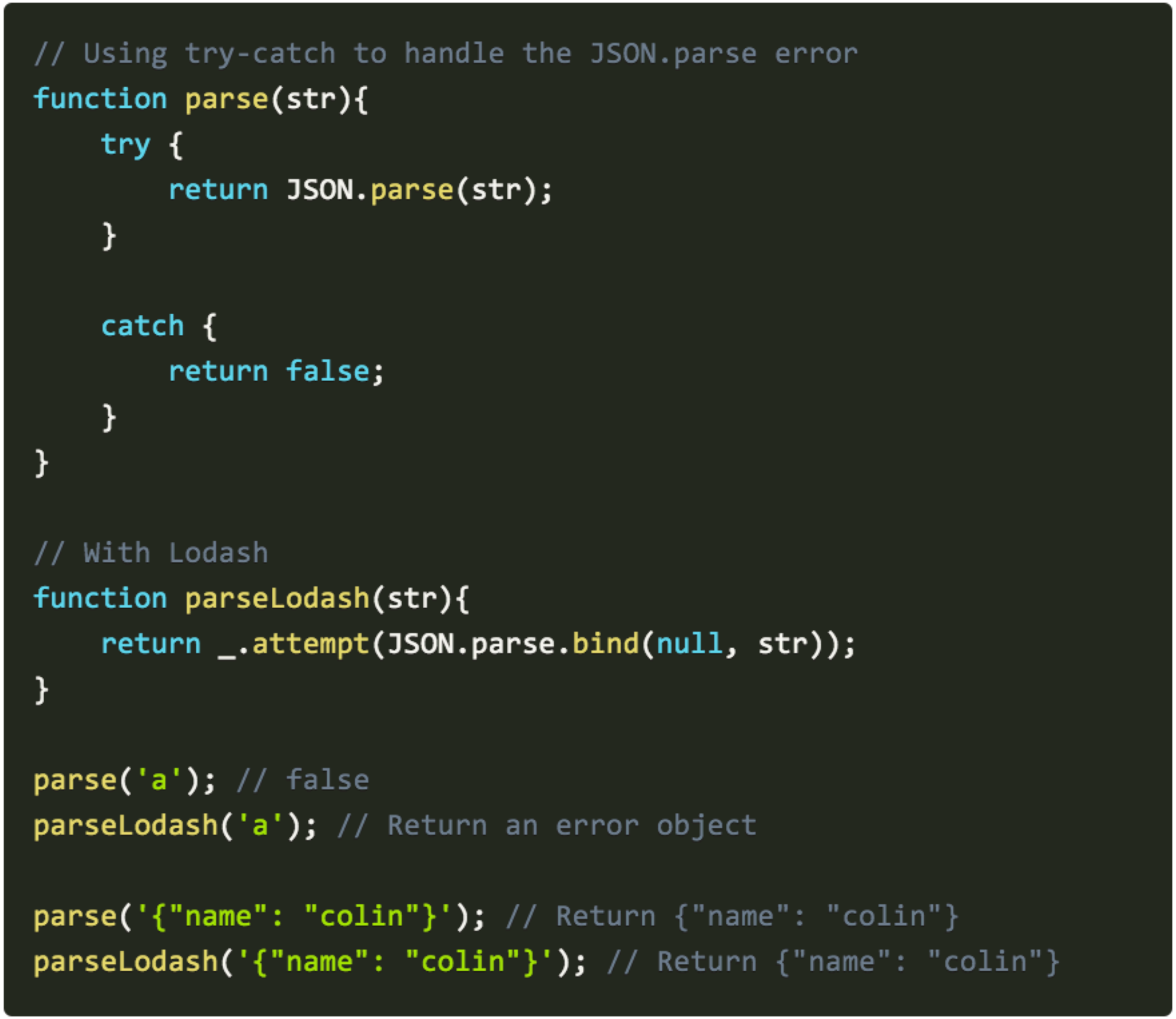
9) Selecting a random item from a list

```
var luckyDraw = ["Colin", "John", "James", "Lily", "Mary"];

function pickRandomPerson(luckyDraw){
    var index = Math.floor(Math.random() * (luckyDraw.length -1))
;
    return luckyDraw[index];
}

pickRandomPerson(luckyDraw); // John

// Lodash
_.sample(luckyDraw); // Colin
```

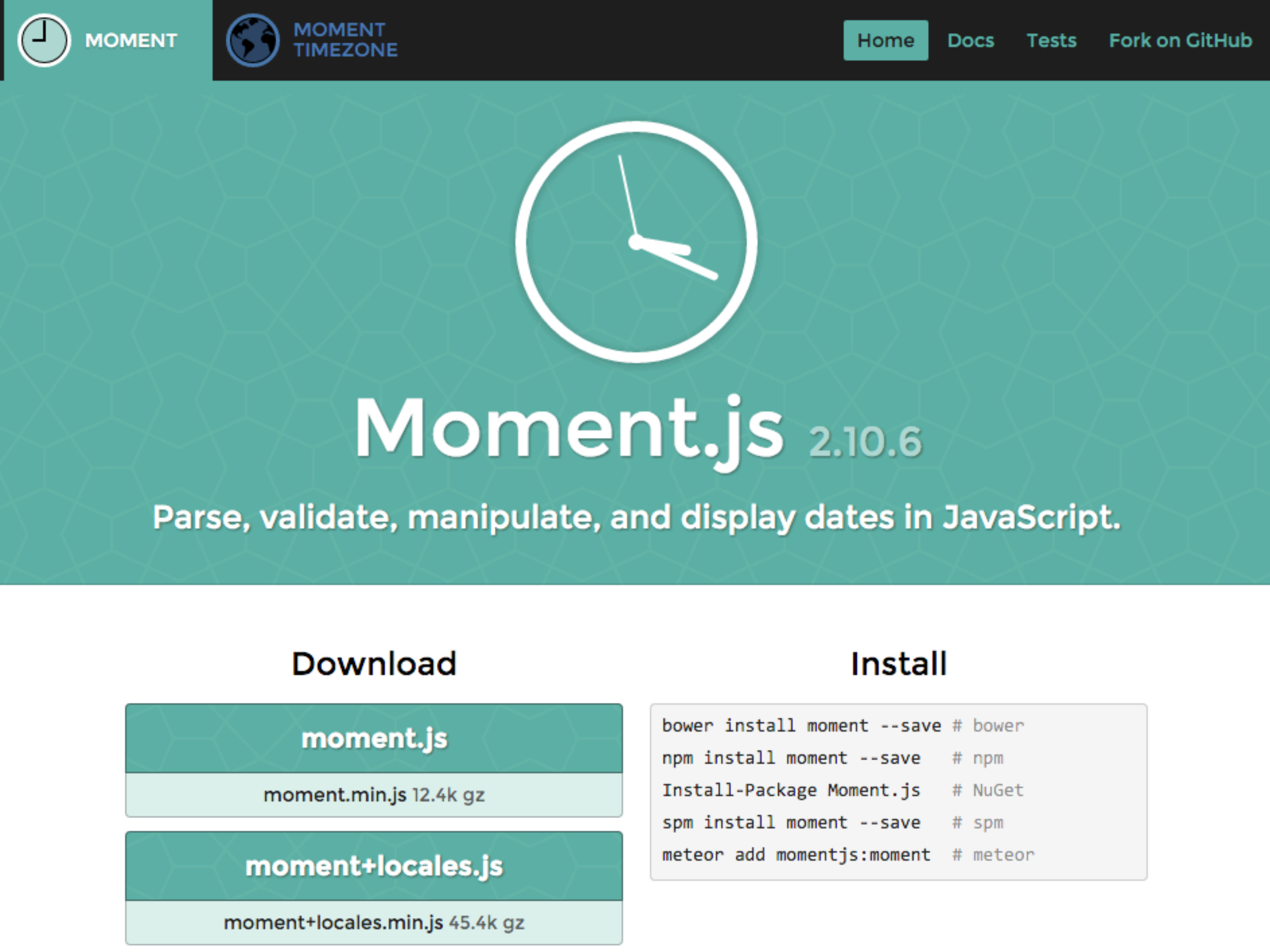
Multiple random item

```
var luckyDraw = ["Colin", "John", "James", "Lily", "Mary"];

// Lodash - Getting 2 random item
_.sample(luckyDraw, 2); // ['John','Lily']
```

10) Error handling for JSON.parse



```
// Using try-catch to handle the JSON.parse error
function parse(str){
    try {
        return JSON.parse(str);
    }

    catch {
        return false;
    }
}

// With Lodash
function parseLodash(str){
    return _.attempt(JSON.parse.bind(null, str));
}

parse('a'); // false
parseLodash('a'); // Return an error object

parse('{"name": "colin"}'); // Return {"name": "colin"}
parseLodash('{"name": "colin"}'); // Return {"name": "colin"}
```



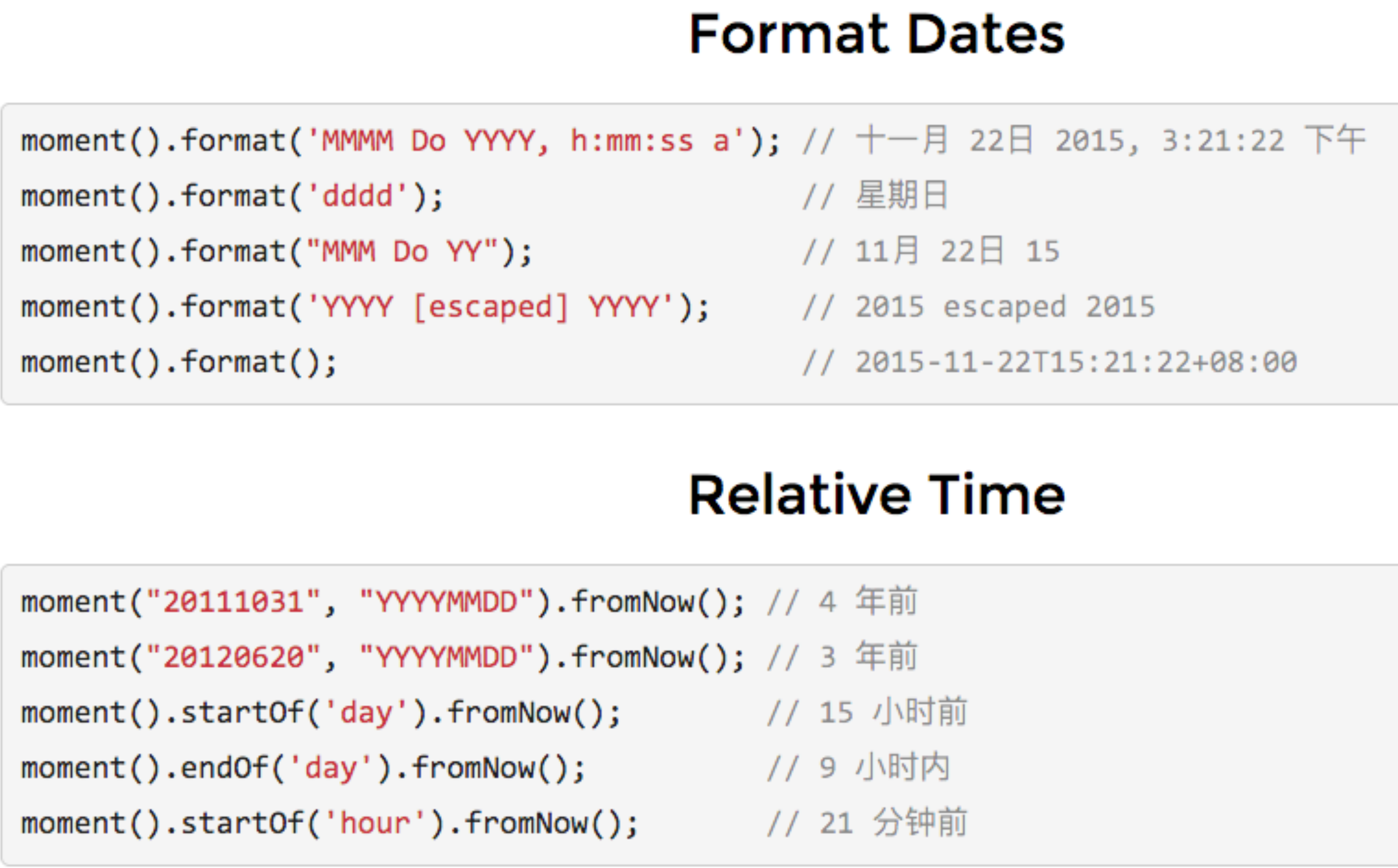
Moment.js 2.10.6

Parse, validate, manipulate, and display dates in JavaScript.

Download

Install

```
bower install moment --save # bower
npm install moment --save   # npm
Install-Package Moment.js   # NuGet
spm install moment --save   # spm
meteor add momentjs:moment  # meteor
```

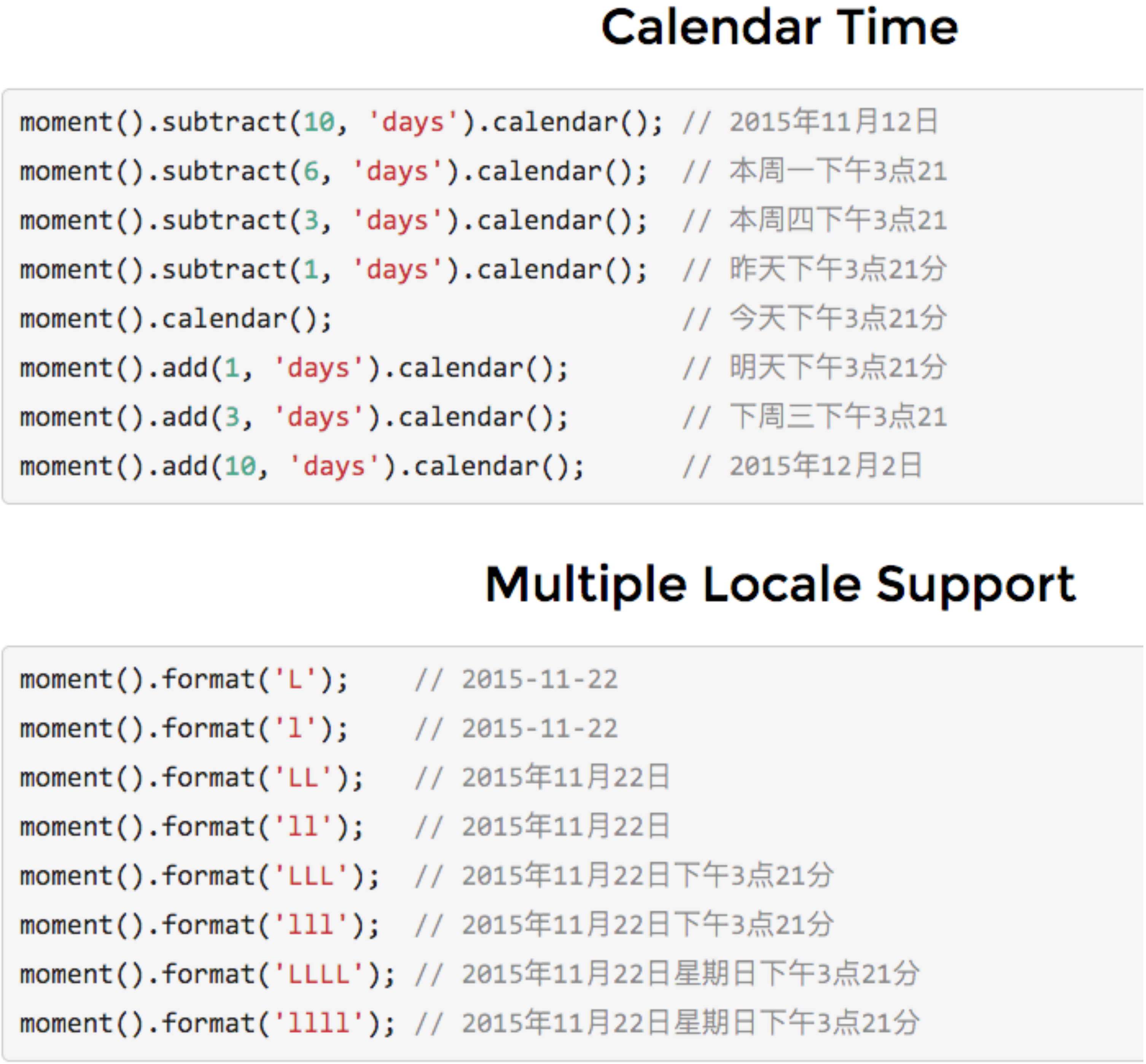


Format Dates

```
moment().format('MMMM Do YYYY, h:mm:ss a'); // 十一月 22日 2015, 3:21:22 下午
moment().format('dddd');                    // 星期日
moment().format("MMM Do YY");               // 11月 22日 15
moment().format('YYYY [escaped] YYYY');     // 2015 escaped 2015
moment().format();                          // 2015-11-22T15:21:22+08:00
```

Relative Time

```
moment("20111031", "YYYYMMDD").fromNow(); // 4 年前
moment("20120620", "YYYYMMDD").fromNow(); // 3 年前
moment().startOf('day').fromNow();        // 15 小时前
moment().endOf('day').fromNow();          // 9 小时内
moment().startOf('hour').fromNow();       // 21 分钟前
```



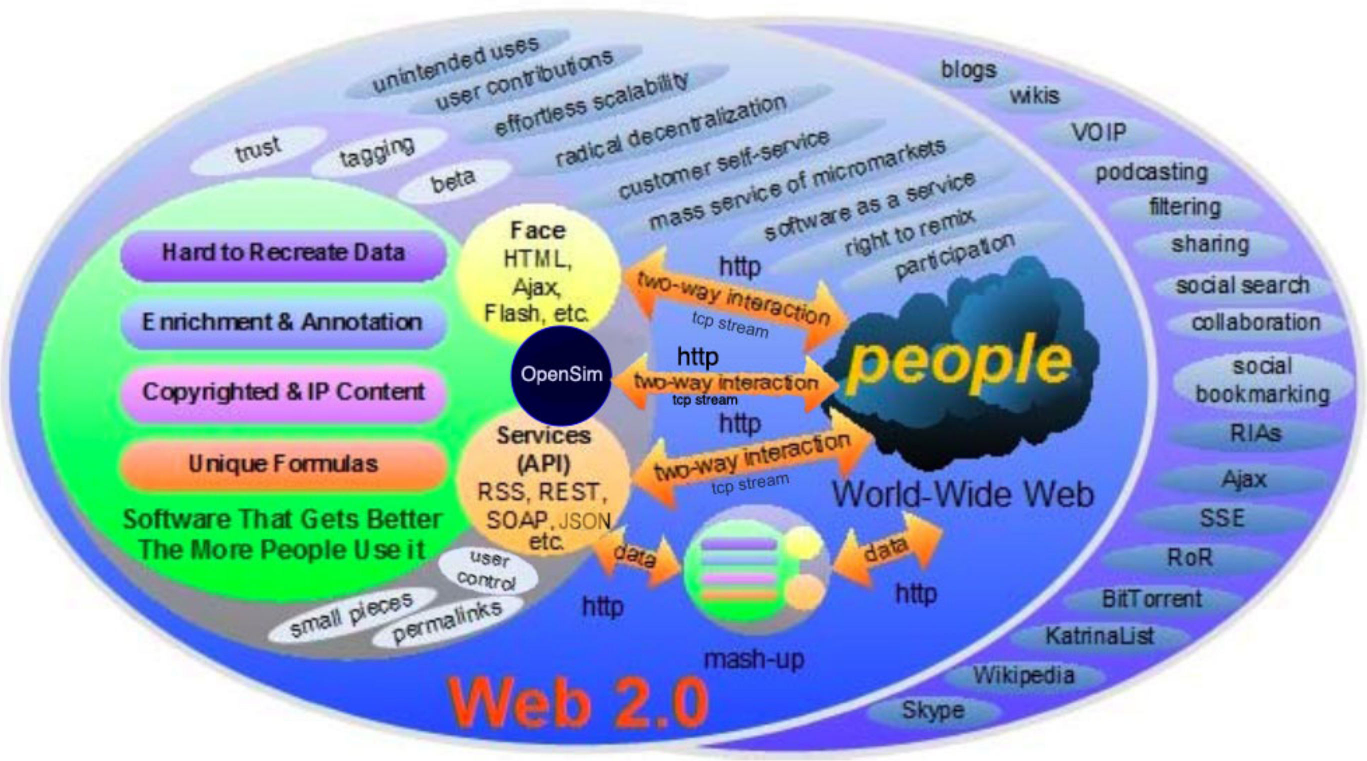
Calendar Time

```
moment().subtract(10, 'days').calendar(); // 2015年11月12日
moment().subtract(6, 'days').calendar();  // 本周一下午3点21
moment().subtract(3, 'days').calendar();  // 本周四下午3点21
moment().subtract(1, 'days').calendar();  // 昨天下午3点21分
moment().calendar();                      // 今天下午3点21分
moment().add(1, 'days').calendar();       // 明天下午3点21分
moment().add(3, 'days').calendar();       // 下周三下午3点21
moment().add(10, 'days').calendar();      // 2015年12月2日
```

Multiple Locale Support

```
moment().format('L');    // 2015-11-22
moment().format('l');    // 2015-11-22
moment().format('LL');   // 2015年11月22日
moment().format('ll');   // 2015年11月22日
moment().format('LLL');  // 2015年11月22日下午3点21分
moment().format('lll');  // 2015年11月22日下午3点21分
moment().format('LLLL'); // 2015年11月22日星期日下午3点21分
moment().format('llll'); // 2015年11月22日星期日下午3点21分
```

Thank you!


unintended uses
user contributions
effortless scalability
radical decentralization
customer self-service
mass service of micromarkets
software as a service
right to remix
participation
trust
tagging
beta
Hard to Recreate Data
Enrichment & Annotation
Copyrighted & IP Content
Unique Formulas
Software That Gets Better The More People Use it
Face HTML, Ajax, Flash, etc.
OpenSim
Services (API) RSS, REST, SOAP, JSON etc.
http
two-way interaction
tcp stream
people
World-Wide Web
data
mash-up
user control
small pieces
permalinks
Web 2.0
blogs
wikis
VOIP
podcasting
filtering
sharing
social search
collaboration
social bookmarking
RIAs
Ajax
SSE
RoR
BitTorrent
KatrinaList
Wikipedia
Skype