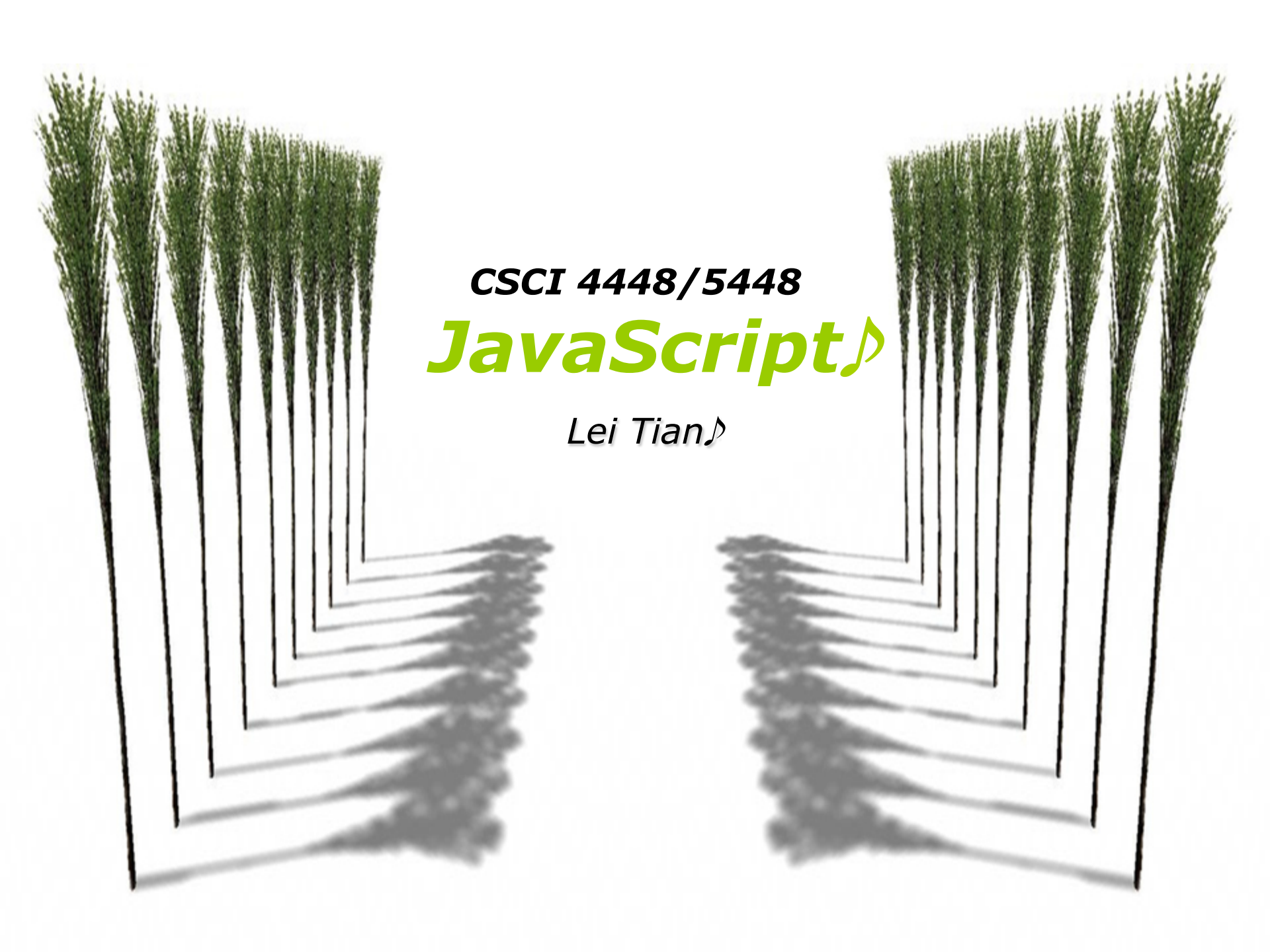


CSCI 4448/5448

JavaScript♪

Lei Tian♪

Table of Contents

Table of Contents

Introduction

History

- History
  - Developed by Brendan Eich
  - Name: Mocha -> LiveScript -> JavaScript
  - Appear since 1996
  - Adopt by ECMA in 1997
  - Become standardized version: ECMAscript
    - **JavaScript = ECMAscript**

Introduction

Definition

- What is JavaScript?
  - A scripting Language
  - An interpreted language (execute without pre-compilation)
  - An object-oriented language
  - Weakly type language
  - Designed to add interactivity to HTML

Introduction

Advantages

- What can a JavaScript do?
  - Give HTML design a programming tool
  - Put dynamic text into HTML page
  - React to events
  - Read and modify HTML elements
  - Detect browser environment
  - Validate information before submission

Features

Dynamics

- Dynamic type
- Object based
- Run-time evaluation

Functional

- First-class function
- Nested function
- Closure

Prototype-based

- Prototypes
- Function as constructor
- Function as method

Let’s We Get Started!

Preparations

Background

- HTML / XHTML language

Edit Tools

- JavaScript editor
- Dreamweaver
- WordPad / NotePad

Debugging Plug-in

- Firebug for Firefox

♪

Get Stated – Using JavaScript

External JavaScript

- External file, eg: “my-script.js”
- <script src="my-script.js" type="text/javascript"></script>
- Define all the variables, objects and functions

Internal JavaScript

- <script type="text/javascript">JavaScript code</script>
- Put in <head>…</head>
  - Call when you want or invoke when it triggered
- Put in <body>…</body>
  - Invoke when page load

Basic Syntax

Variable

- Case sensitive
  - “var a”, “var A” are two different variables
- Declare variable
  - Keyword: **var**, eg: var x
- Dynamic type
  - **type associate with value, no type declaration**
- Scope
  - **Local:** within the function
  - **Global:** declare out of function, all page could access

Type of Data

Numeric

- Integer: -2^53 ~ 2^53
- Float

String

- Define by “” or ‘’
- Escape Character “\”, same as C and Java

Boolean

- True, False

Array

♪

Array

Declare

- var myArray = new Array() / new Array(6);

Assign Value

- var myArray = new Array("Tom",12);
- var myArray = ["Tom", 12, "Mike"];
- myArray[1] = "Mike";

Multiple Dimensions

- var personal = new Array();
- Personal[0] = new Array();

A Object, A Container!
Like Java!

Elements could be
different types

Array

Can be sparse

- var myArray = new Array() ;
- myArray(0) = 12; myArray(100) = 15;
- others will be **undefined**

Methods

- pop, push, join, reverse, sort, split, etc.
- Array can be **resized** and **modified**

Regular Objects as Arrays

- number as index of properties

Operations

Numeric

- +, -, *, /, %, ++, --, +=, -=, *= etc

String

- +: string merge or append with string/number

Boolean

- ==, !=, >, >=, <, <=

Conversion

- parseInt()/parseFloat()
- isNaN() – check conversion

Conditional and Loops

If/Else

- Not strict True / False like Java
- False: false, null, undefine, “”, 0, NaN

For in Loops

- For array, value are index not array value
- For object, value are property

nearly identical to Java

Switch

- “Case” could be an expression
- Value need not be int

♪

Functions

A function is an object

- A function could have objects, properties
- Type: Normal function, Anonymous function

Normal Function

- Declaration: keyword: **function**
  - function functionname (param1, param2, …, paramN){…}
- Call/Invocation:
  - functionname (param1, param2, …, paramN);
- Could have return value but not necessity

```
function sayHi(toWhom){
    alert("Hi "+toWhom);
}
sayHi("Tom!");
```

Functions

Anonymous Function

- Capture local variables inside the function
- Constructor/Declaration:

1. Function Literals

```
var hifunc = function(toWhom){
    alert("Hi "+toWhom);
}
hifunc("Tom!");
```

Param

Function Code

2. Constructor

```
var hifunc = new Function("toWhom", "alert("Hi "+toWhom);");
```

Functions – args number

- Could call function with any number of argument

- Fixed Number
  - args fewer, extra agrs are **"undefined"**
  - typeof args == "undefined" to check

- Arbitrary Number
  - Discover number by **arguments.length**
  - Access by **arguments[i]**

/*varargs*/
tell arbitrary number

```
function assignValue(/*varargs*/){
    var num = arguments.length;
    var secondstr = arguments[num-1];
}
assignValue("aa","bb","cc","dd");
```

Functions

Differences from Java

- Could have global functions
- Functions are first-class datatypes
- Don’t need declare return type
- Pass multiple types of parameters
- Could supply any number of arguments
- Create anonymous functions

Functions are Objects

Dynamic types

Functions - Recall

Location

- External function: in extern file - .js file
- Internal function: in current file - .html file
  - In header: execute when called – mostly event function
  - In body: execute when page load that

Lifetime

- Create once page load to it
- Destroy when page is destroyed

Objects

- **JavaScript is an Object-Oriented Programming Language !!!**

- **Properties**
  - Values associated with the objects
  - ```
    var txt = "Hellow world";
    document.write(txt.length);
    ```

- **Methods**
  - Actions performed on the objects
  - ```
    var txt = "Hellow world";
    document.write(txt.toUpperCase());
    ```

♪

Objects

Build-In Objects

- String
- Date

```
var today = new Date();
var d1 = new Date("October 13, 1975 11:13:00");
var d2 = new Date(79,5,24);
var d3 = new Date(79,5,24,11,33,0);
```

- Array
- Math

```
var x=Math.PI;
var y=Math.sqrt(16);
document.write(Math.random());
document.write(Math.round(4.7));
```

- RegExp - **Regular Expression**
  - An object that describes a pattern of characters
  - Function: Search + Replace

Basics for Objects

Constructor

- Function name is class name; keywords: **new**
- Property define must use **this**

Properties

- Can be create and refer in outside code

Methods

- Properties whose value are function

♪

Basics for Objects

Example

```
function newClass(a){
    this.foo = a;
    this.printName = function(){
        return "Class Name is newClass");
        };
}
 m = new newClass(10);
 m.bar = 100;
 m.prinntName();
```

Property

Method

Now, m has two properties:
m.Foo = 10;
m.Bar = 100;

Prototype for Objects

Function

- Save space – each instance needn't copy function
- Inheritance

```
function Circle(radius) {
    this.radius = radius;
    Circle.prototype.getArea = function() {
        return(Math.PI * this.radius * this.radius);
        };
}
var c = new Circle(10);
c.getArea();
```

No need to copy the function

```
function ClassA()
{
    this.a='a';
}
function ClassB()
{
    this.b='b';
}
ClassB.prototype=new ClassA();
var objB=new ClassB();
alert(objB.a);
ClassB.prototype.a='changed!!';
alert(objB.a);
```

Class B will have all the properties of Class A

Rolling Your Own Namespace

Idea

- Have related functions that do not use object properties
- Put together, call with Utils.func1, Utils.func2 …

Syntax

1. A object with no constructor;
2. Assign functions as properties
3. Call object with function
   - Var Utils = {}; / var Utils = new Object();
     Utils.foo = function(m){…}
     Utils.bar = function(m, n){…}
     Utils.foo(4);
     Utils.bar(4,5);

JSON - JavaScript Object Notation

Definition

- A simple textual representation of JavaScript objects
  - **var jsonObject = {**
    property1: value1,
    property2: value2,
    … … }

Values could be another JSON object or a function

Retrieve

- jsonObject. property*i*. subproperty*m*. …

Convert / Parse

- var myObject = eval('(' + myJSONtext + ')');
- var myObject = JSON.parse(myJSONtext, reviver);

Filter and replace function

♪

Example for JSON

```
var user =
    {
        "username":"andy",
        "age":20,
        "info": { "tel": "123456", "cellphone": "98765"},
        "address":
            [
                {"city":"beijing","postcode":"222333"},
                {"city":"newyork","postcode":"555666"}
            ]
    }
alert(user.username);
alert(user.age);
alert(user.info.cellphone);
alert(user.address[0].city);
alert(user.address[0].postcode);

var str = '{ "name": "Violet", "occupation": "character" }';

var obj = eval('(' + str + ')');
alert(obj.toJSONString());

var obj2 = str.parseJSON();
alert(obj2.toJSONString());
```

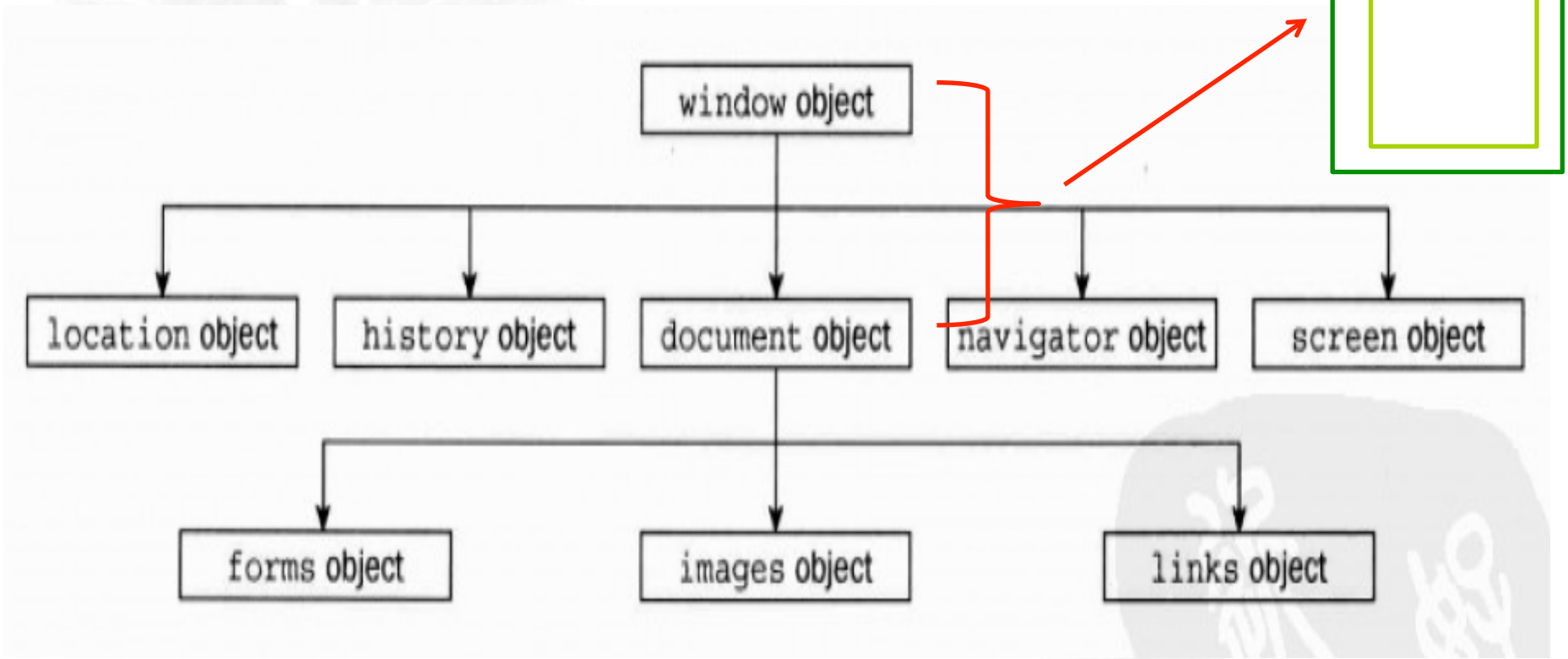
Retrieve value

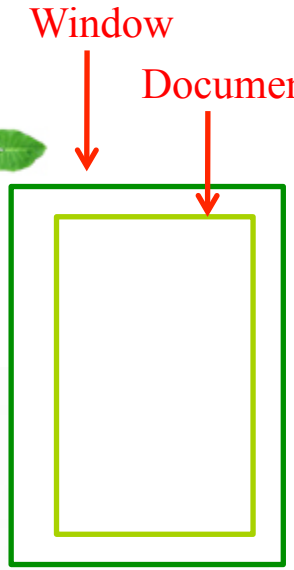
String -> JSON

JSON -> String

Advanced JavaScript

Browser Object Model (BOM)





BOM

Window

- The frame/window of browser
- Access window size, scan history, navigator info and etc
- Global variable → No need to write object
  - alert("Hello!"); / defaultStatus = "Hello World"; → In fact, window.alert(); window.defaultStatus();

Location

- Access through window.location / location
- Properties: href, hostname, port, protocal
- Ex: location.href = "myPage.htm"
  location.replace("myPage.htm");

BOM

Navigator

- Detect browser type, serve appropriate infomation
- Contain browser type, name, version and etc
- Properties: appName, appVersion, cookieEnabled and etc

Document

- The webpage, all the display content could be set
- document.write(); document.bgcolor;
- Object Arrays: document.link[]; document.button[];
  document.images[];

What's Next To Learn?

HTML DOM

- A standard way accessing and manipulate HTML document
- Platform and language independent

jQuery

- JavaScript library
- Call 3rd party function – simplify Java programming

AJAX

- Asynchronous JavaScript and XML
- Exchange data with server, update part without the whole page
- Ex: Google map, Gmail, Youtube

References

Website:

- http://www.w3schools.com/js/
- http://www.javascriptkit.com/jsref/
- http://www.ecmainternational.org/publications/standards/Ecma-262.htm

Books

- ***JavaScript the Definitive Guide***
  - **By David Flanagan, O'Reilly.**
  - **The only really complete reference on the JavaScript language. Thorough and well-written**
- ***JavaScript: The Good Parts***
  - **By Douglas Crockford (of JSON and YUI fame), O'Reilly**
  - **Outstanding advanced guide to best practices in core JavaScript**

Questions?

Thank You !