JavaScript, BOM, DOM

Course 2

JavaScript

JavaScript

- Nothing related to Java

JavaScript

- Netscape 2
- by Brendan Eich in 10 days


Firefox
wip

JScript

- IE 3.0
- Compatible with Netscape’s JavaScript

JavaScript

- The only language can run in web(browser)

- VBScript is not support anymore
- Dart is not plan to support in V8

JS? ES?

- 1996 Netscape send JavaScript to ECMA
- 1997 ECMAScript 1.0 published

ECMAScript

JavaScript

Language Spec

Implementation

ECMAScript

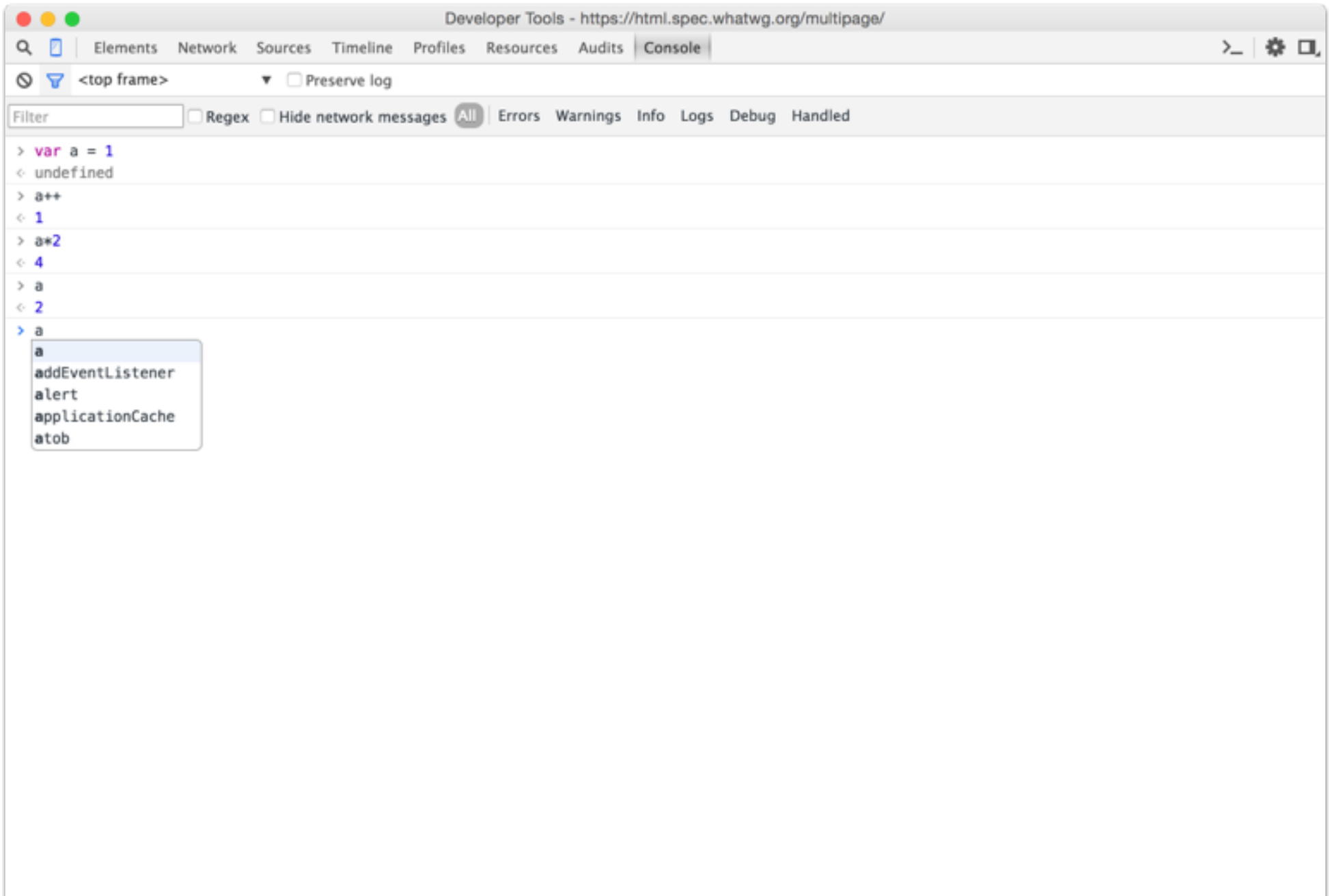
- Latest version is 6(2015)
- Browsers are during 5.1 to 6 process

3	IE6 implements this	1999
4	Abandoned	
5	Fix lots of old issues	2009
5.1	Align 5 to 3	2011
6	Just released	2015

- Not to talk syntax today
- Let’s talk about characteristics

- Type
- Scope/Closure
- Prototype
- Inheritance
- Timer
- this
- Strict Mode

Dev Tool to Play



Type

- JavaScript is weakly typed(untyped)

- No type declaration
- One variable can assign multiple type value

```
var foo = 1;

foo = 'string';

foo = true;

foo = [1, 2, 3];

foo = {
  bar: 1
};
```

Float

- IEEE-754

- `0.1 + 0.2 = 0.30000000000000004`

Special Number

- Infinity
- NaN

NaN

- Not a Number
- Produced when type conversion fail

```
typeof NaN == 'number'
```

Compare

- == and ===

==

- Equality, with type conversion

```
'' == 0

0 == '0'

'' != '0'

[] == false

NaN != NaN
```

Zeros in JavaScript

zero.milosz.ca

2013-11-05:

A kind fellow named Ilya informed me that `{}+foo` is probably parsed by JavaScript as `{};+foo`. To mitigate the issue, all operations are now being wrapped in parentheses before evaluation (e.g. `({}+foo)`.)

==	0	'0'	[0]	[]	{}	''	null	[null]	undefined	NaN	[NaN]	'a'	Infinity	1	'1'	[1]	true	false	'true
0	true	true	true	true	false	true	false	true	false	false	false	false	false	false	false	false	false	true	fals
'0'	true	true	true	false	false	false	false	false	false	false	false	false	false	false	false	false	false	true	fals
[0]	true	true	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	true	fals
[]	true	false	false	false	false	true	false	false	false	false	false	false	false	false	false	false	false	true	fals
{}	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	fals
''	true	false	false	true	false	true	false	true	false	false	false	false	false	false	false	false	false	true	fals
null	false	false	false	false	false	false	true	false	true	false	false	false	false	false	false	false	false	false	fals
[null]	true	false	false	false	false	true	false	false	false	false	false	false	false	false	false	false	false	true	fals
undefined	false	false	false	false	false	false	true	false	true	false	false	false	false	false	false	false	false	false	fals
NaN	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	fals
[NaN]	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	fals
'a'	false	false	false	false	false	false	false	false	false	false	false	true	false	false	false	false	false	false	fals
Infinity	false	false	false	false	false	false	false	false	false	false	false	false	true	false	false	false	false	false	fals
1	false	false	false	false	false	false	false	false	false	false	false	false	false	true	true	true	true	false	fals
'1'	false	false	false	false	false	false	false	false	false	false	false	false	false	true	true	true	true	false	fals
[1]	false	false	false	false	false	false	false	false	false	false	false	false	false	true	true	false	true	false	fals
true	false	false	false	false	false	false	false	false	false	false	false	false	false	true	true	true	true	false	fals
false	true	true	true	true	false	true	false	true	false	false	false	false	false	false	false	false	false	true	fals
'true'	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	tru
'false'	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	fals

===	0	'0'	[0]	[]	{}	''	null	[null]	undefined	NaN	[NaN]	'a'	Infinity	1	'1'	[1]	true	false	'true
0	true	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	fals

- Identity, no type conversion

- Always use === instead of ==

NaN

```
NaN != NaN

NaN !== NaN

isNaN(NaN) === true
```

created by Brian Leroux & Andrew Lunny. sparodically uncurated by David Trejo.

regular expression and slash

jan 29 , 2014

When I use regular expressions and I want to validate a range of letters, I can do it using `a-z` or `A-Z`. Even when I use `A-z` it works fine too. The problem comes doing some test:

```
/[A-Z]/.test("A"); // true
/[A-Z]/.test("b"); // false
/[A-Z]/.test("Z"); // true
/[A-Z]/.test("z"); // false
/[a-z]/.test("a"); // true
/[a-z]/.test("A"); // false
/[a-z]/.test("z"); // true
/[a-z]/.test("Z"); // false
```

The weird thing comes when I do this test:

```
/[A-z]/.test("A"); // true
/[A-z]/.test("a"); // true
/[A-z]/.test("Z"); // true
/[A-z]/.test("z"); // true
/[A-z]/.test("m"); // true
/[A-z]/.test("D"); // true
/[A-z]/.test("\\"); // true WTF?
```

It's supposed to accept only letters from `A to Z` and `a to z`. Can someone explain this?

Variable Scope

- Global
- Local

```
function fff() {
  foo = 1;

  var bar = 2;
}

fff();

foo; // 1
bar; // ReferenceError
```

- Without var, foo is global variable

- Without var, foo is global variable
- Always use var

```
function fff() {
  foo = 1;

  var bar = 2;
}

fff();

foo; // 1
bar; // ReferenceError

window.foo; // 1
```

window

- The global object
- All global variables attached on it
- Root of everything

Closure

```
var gnum = 1;

function fff() {
  var bar = 2;

  var ggg = function () {
    var foo;

    gnum; // 1
    bar; // 2
  };

  ggg();
}

fff();
```

Variable Lookup

1. Look context's variables
2. If found, return the variable
3. If not found, look for parent context
4. If not have parent, throw ReferenceError
5. If have parent, set parent as context,
   goto step 1

notice

- Only function scope
- No block scope until ES6
- Block scope variable uses `let` to declare

Dev Tool Demo

Class

- No real class until ES6
- Use new to create instance

Constructor

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}
```

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}
```

```
var a = new Point(1, 2);

a.x; // 1
```

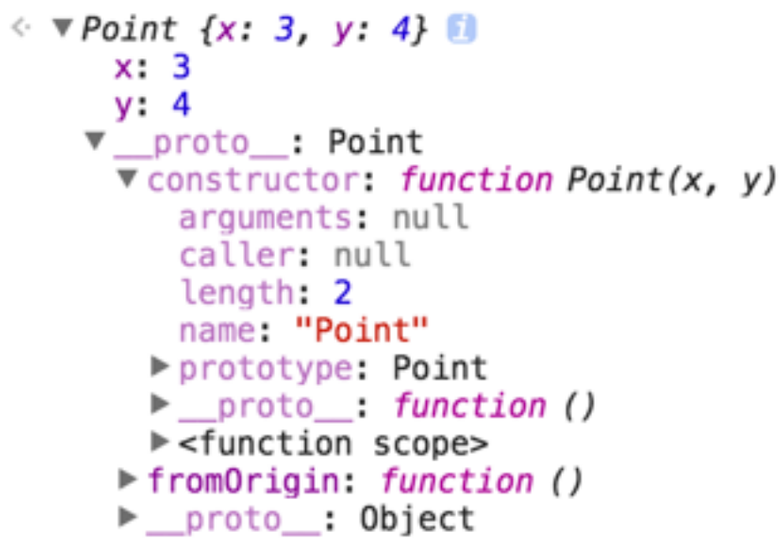
```
function Point(x, y) {
  this.x = x;
  this.y = y;

  this.fromOrigin = function () {
    return Math.sqrt(this.x * this.x + this.y * this.y);
  };
}

var a = new Point(3, 4);

a.fromOrigin(); // 5
```

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

Point.prototype.fromOrigin = function () {
  return Math.sqrt(this.x * this.x + this.y * this.y);
};

var a = new Point(3, 4);

a.fromOrigin(); // 5
```

Property Lookup

1. Look target's properties
2. If found, return the property
3. If not found, look for prototype
4. If not have prototype, return undefined
5. If have prototype, set prototype as target, goto step 1

Property Lookup



Inheritance

```
function Point3(x, y, z) {
  Point.call(this, x, y);

  this.z = z;
}
```

```
function Point3(x, y, z) {
  Point.call(this, x, y);

  this.z = z;
}
Point3.prototype = new Point();
```

```
function Point3(x, y, z) {
  Point.call(this, x, y);

  this.z = z;
}
Point3.prototype = new Point();

Point3.prototype.fromOrigin = function () {
  return Math.sqrt(this.x * this.x + this.y * this.y + this.z * this.
};

var a = new Point3(1, 2, 3);
```

Object.create

```
function Point3(x, y, z) {
  Point.call(this, x, y);

  this.z = z;
}
Point3.prototype = Object.create(Point.prototype);

Point3.prototype.fromOrigin = function () {
  return Math.sqrt(this.x * this.x + this.y * this.y + this.z * this.
};

var a = new Point3(1, 2, 3);
```

Object.create

- IE6–8 not available

Class

```
class Point {
  constructor(x, y) {
    this.x = x;
    tihs.y = y;
  }
};
```

```
class Point {
  constructor(x, y) {
    this.x = x;
    tihs.y = y;
  }
};

class Point3 extends Point {
  constructor(x, y, z) {
    super();
    this.z = z;
  }
};
```

Class

- Only ES6 supports

setTimeout

- Calls a function or executes a code snippet after a specified delay

```
setTimeout(function () {
  alert('ha');
}, 1000);
```

setInterval

- Calls a function or executes a code snippet repeatedly
- with a fixed time delay between each call to that function

```
setInterval(function () {
  alert('ha');
}, 1000);
```

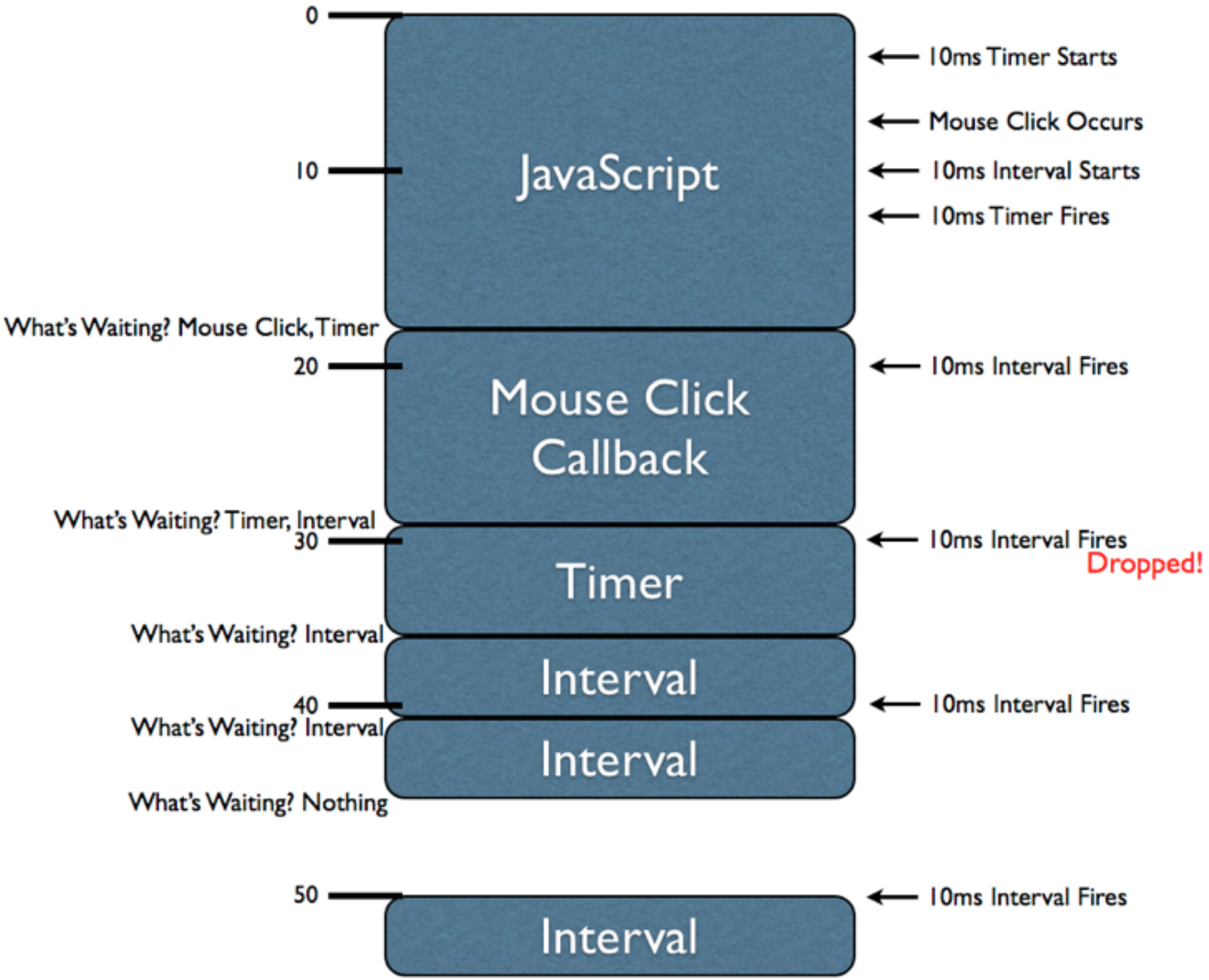


Image: http://ejohn.org/blog/how-javascript-timers-work/

Block

- The browser UI interaction is also queued in JavaScript timer
- If a function didn’t stop, the browser blocks

Understand Timer

- Write unblocking code
- Heavy computing job
  - setTimeout to calculate in several block
- Help debug UI blocking issue
  - UI change uses the same timer

this

- Lets look at another slide

this

othree@JSDC

Strict Mode

- Force use good part of JavaScript

```
function foo() {
  'use strict';

  // strict JavaScript code
}
```

Strict Mode

- Always require `var`
- No ambiguous `this`

```
4          142;
```

Simplifying variable uses

Strict mode simplifies how variable names map to particular variable definitions in the code. Many compiler optimizations rely on the ability to say that variable *X* is stored in *that* location: this is critical to fully optimizing JavaScript code. JavaScript sometimes makes this basic mapping of name to variable definition in the code impossible to perform until runtime. Strict mode removes most cases where this happens, so the compiler can better optimize strict mode code.

First, strict mode prohibits `with`. The problem with `with` is that any name inside the block might map either to a property of the object passed to it, or to a variable in surrounding (or even global) scope, at runtime: it's impossible to know which beforehand. Strict mode makes `with` a syntax error, so there's no chance for a name in a `with` to refer to an unknown location at runtime:

```
"use strict";
var x = 17;
with (obj) // !!! syntax error
{
  // If this weren't strict mode, would this be var x, or
  // would it instead be obj.x?  It's impossible in general
  // to say without running the code, so the name can't be
  // optimized.
  x;
}
```

The simple alternative of assigning the object to a short name variable, then accessing the corresponding property on that variable, stands ready to replace `with`.

Second, `eval` of strict mode code does not introduce new variables into the surrounding scope. In normal code `eval("var x;")` introduces a variable `x` into the surrounding function or the global scope. This means that, in general, in a function containing a call to `eval` every name not

- JavaScript is just a programming language

- To talk to browser, we need APIs

- BOM to Browser

- DOM to Document(HTML)

BOM

Browser Object Model

Talk to Browser

- Window
- Navigator
- Document
- Cookie
- History
- Web APIs

Window

- The root object
- Represent the browser window
- System information

```
window.document;

window.screen.width;

window.screenX;

window.open();
```

Navigator

- Browser information
- Some Web API

```
navigator.userAgent;

navigator.oscpu;

navigator.platform;
```

Document

- Document Object, DOM

Cookie

- The only? thing can store at client side
- Important for identity

Identity

- User login session
- Device identification (GA)

Login Flow

- Login to server
- Server generate session id
- Store session id in cookie

Revisit Flow

- Link to server
- Take session id to server

Important

- Don’t store sensitive data
- ex: password

Write Cookie

```
document.cookie = "name=oeschger";

document.cookie = "favorite_food=tripe";
```

Read Cookie

```
document.cookie;

// "name=oeschger; favorite_food=tripe"
```

Use Lib to Deal With

Write a new cookie

```
document.cookie = newCookie;
```

In the code above, `newCookie` is a string of form *`key=value`*. Note that you can only set/update a single cookie at a time using this method. Consider also that:

- Any of the following cookie attribute values can optionally follow the key-value pair, specifying the cookie to set/update, and preceded by a semi-colon separator:
  - `;path=`*`path`* (e.g., `'/'`, `'/mydir'`) If not specified, defaults to the current path of the current document location.

    Note: Prior to Gecko 6.0, paths with quotes were treated as if the quotes were part of the string, instead of as if they were delimiters surrounding the actual path string. This has been fixed.

    The path must be *absolute* (see RFC 2965). For more information on how to use relative paths, see this paragraph.
  - `;domain=`*`domain`* (e.g., `'example.com'`, `'.example.com'` (includes all subdomains), `'subdomain.example.com'`) If not specified, defaults to the host portion of the current document location.
  - `;max-age=`*`max-age-in-seconds`* (e.g., `60*60*24*365` or `31536e3` for a year)
  - `;expires=`*`date-in-GMTString-format`* If not specified it will expire at the end of session.
    - See `Date.toUTCString()` for help formatting this value.

    When user privacy is a concern, It is important that any web app implementation will invalidate cookie data after a certain timeout and won't rely on the browser clearing session cookies

    One of the most beloved features of Firefox prevents session cookies from ever expiring .

    The same issue is also occuring with google chrome (and probably with other browsers offering similar features)
  - `;secure` (cookie to only be transmitted over secure protocol as https)
- The cookie value string can use `encodeURIComponent()` to ensure that the string does not contain any commas, semicolons, or whitespace

History

- Browser history
- The back button

Back

```
history.back();

history.go(-1);
```

Forward

```
history.forward();

history.go(1);
```

Add History Entry

```
history.pushState("/path/to/target");
```

DOM

Document Object Model

DOM

- Interactive with HTML document
- With a tree data structure

DOM Tree

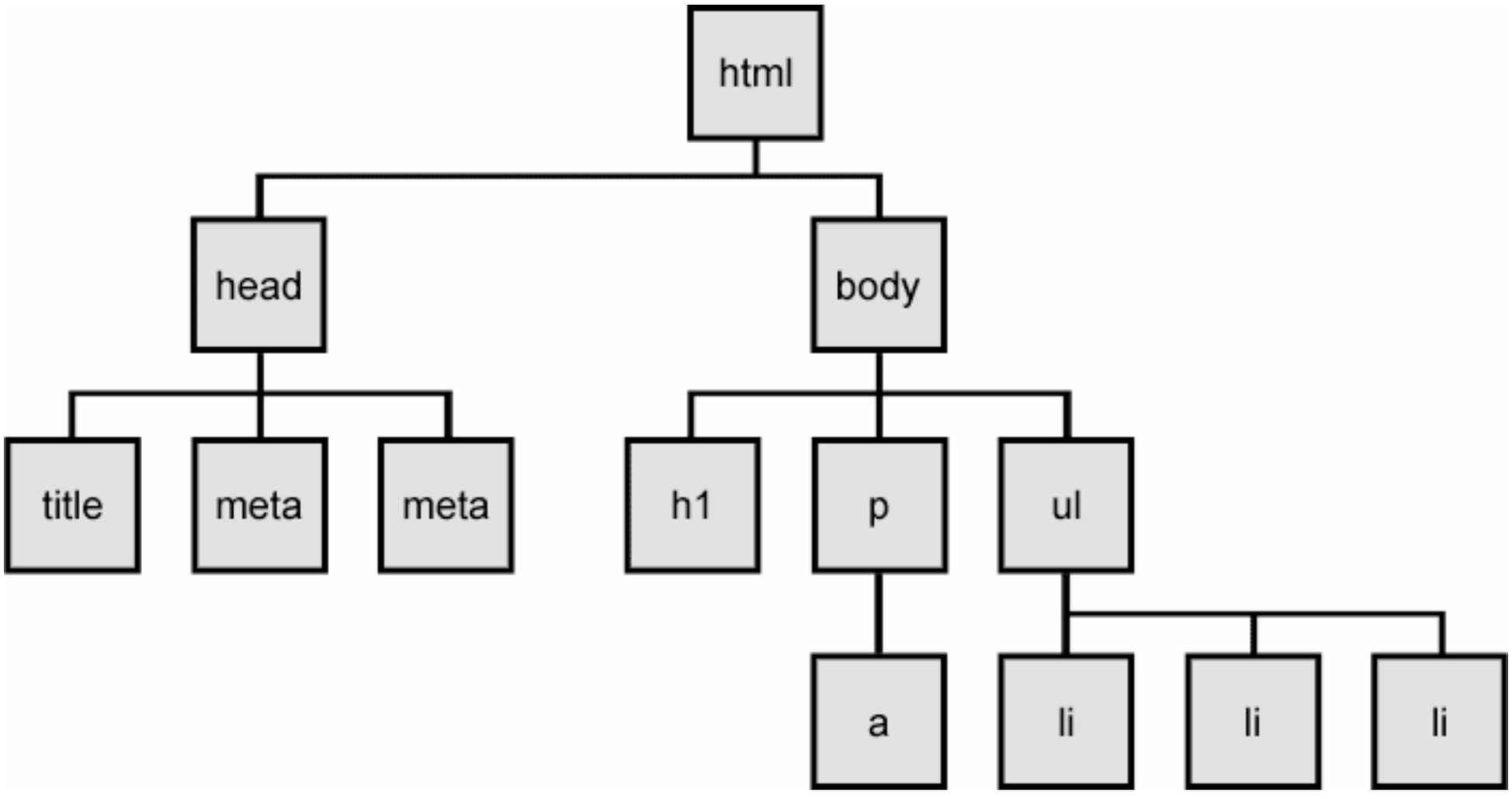
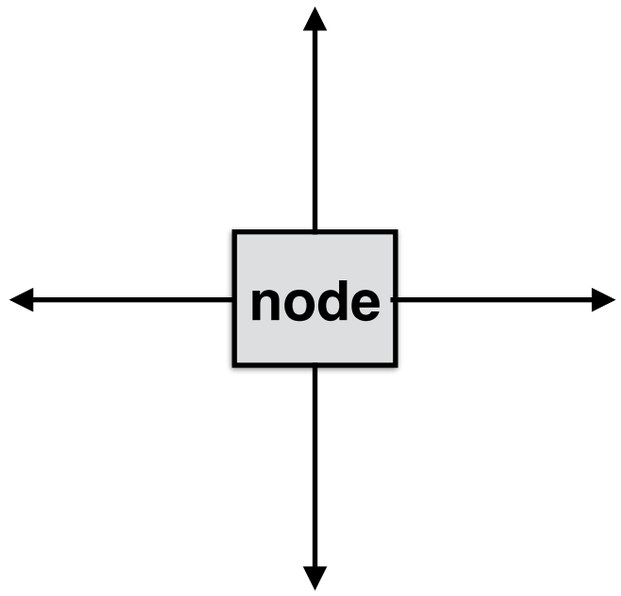


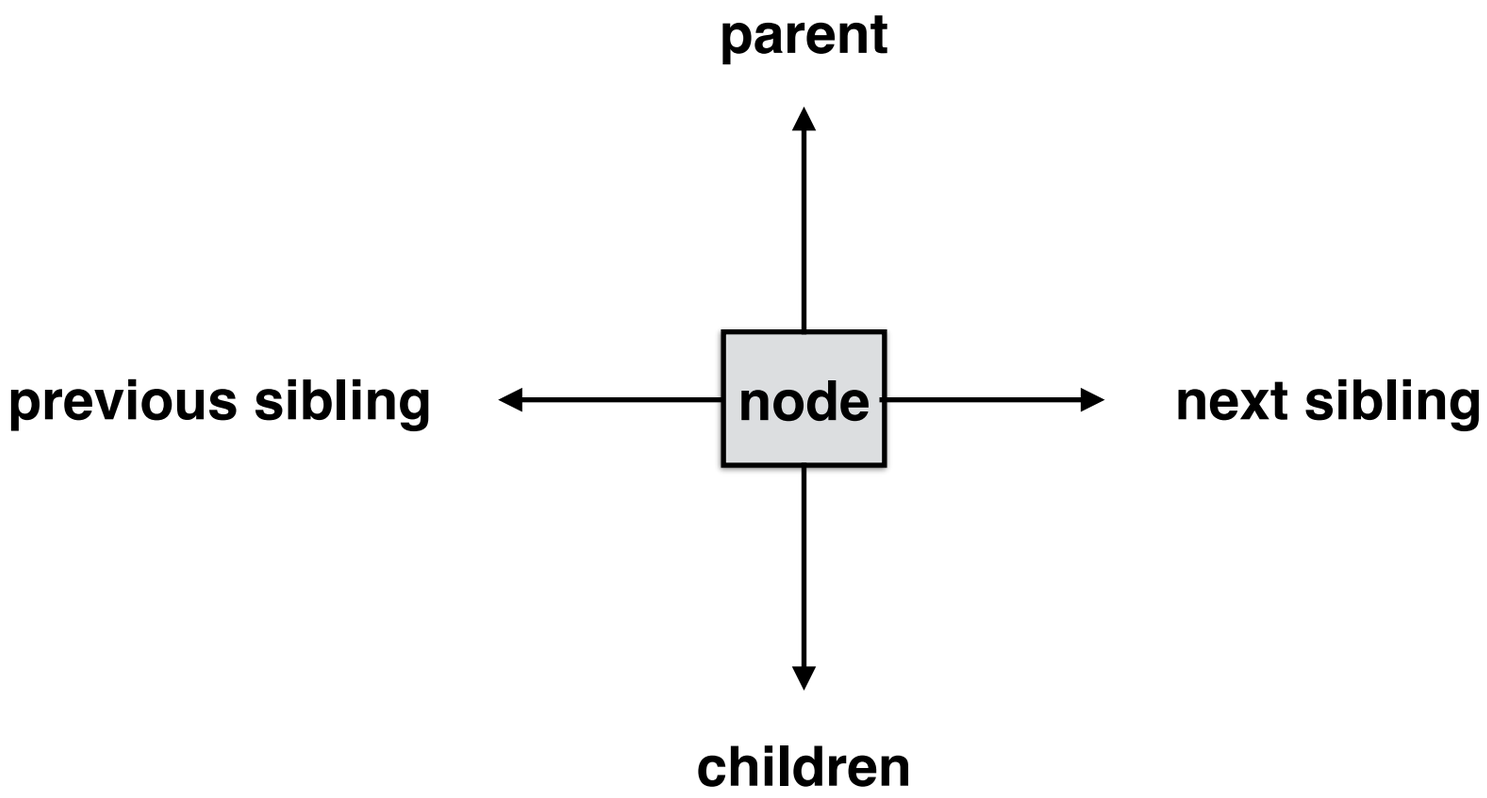
Image: http://www.webstepbook.com/supplements/slides/lecture16-dom_tree.shtml

Flow

- Find the node(traversing)
- Do some change

DOM Traversing




parent
previous sibling
node
next sibling
children

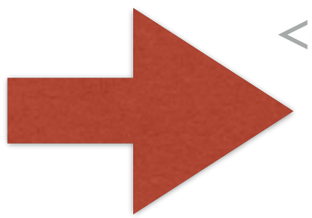
```
node.parentNode; // node

node.nextSibling; // node

node.previousSibling; // node

node.firstChild; // node

node.children; // node list
```

Start Point

- The document object

```
document; // root node

document.children; // node

document.getElementById("target"); // node #target
```

```
<html>
  <body>
    <div>
      <article id="target"></article>
      <article></article>
      <article></article>
    </div>
  </body>
</html>
```



```
<html>
  <body>
    <div>
      <article id="target"></article>
      <article></article>
    </div>
  </body>
</html>
```

```
document; // root node

document.children; // node list

document.getElementById("target"); // node #target

document.getElementsByClassName("target"); // node list .target
```

```
<html>
  <body>
    <div>
      <article class="target"></article>
      <article></article>
      <article class="target active"></article>
      <article></article>
      <article></article>
    </div>
  </body>
</html>
```

```
<html>
  <body>
    <div>
      <article class="target"></article>
      <article></article>
      <article class="target active"></article>
      <article></article>
      <article></article>
    </div>
  </body>
</html>
```

NodeList

- Array like object

```
for (var i = 0; i < myNodeList.length; ++i) {
  var item = myNodeList[i];
}
```

```
for (key in myNodeList) {
  // dont do this
  // also enumerate length
}
```

to Array

```
Array.prototype.slice.call(nodeList);
```

```
document; // root node

document.children; // node list

document.getElementById("target"); // node #target

document.getElementsByClassName("target"); // node list .target

document.getElementsByTagName("article"); // node list article
```

Edit Node

- Edit children
- Edit attribute
- innerHTML

Append

```
var insertedNode = parentNode.appendChild(newNode);
```

Remove

```
var oldChild = element.removeChild(child);
```

Insert

```
parentNode.insertBefore(newNode, referenceNode);
```

Set Attribute

```
elementNode.setAttribute(name, value);
```

Get Attribute

```
var attribute = elementNode.getAttribute(attributeName);
```

Nodes are From

- Query from document/other doc
- Created by JS

```
<p></p>
```

```
<p>This is text</p>
```

Add Text

```
var textNode = document.createTextNode("This is text”);

var pNode = document.querySelector("p");

pNode.appendChild(textNode);
```

Yes, Text is Node

```
<ul>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

```
<ul>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

Issues

- Hard to deep into target
- White space

One More Thing

- Attach event listener

Events

- All user inputs
- Browser processing events
- Message events
- …

Click Event

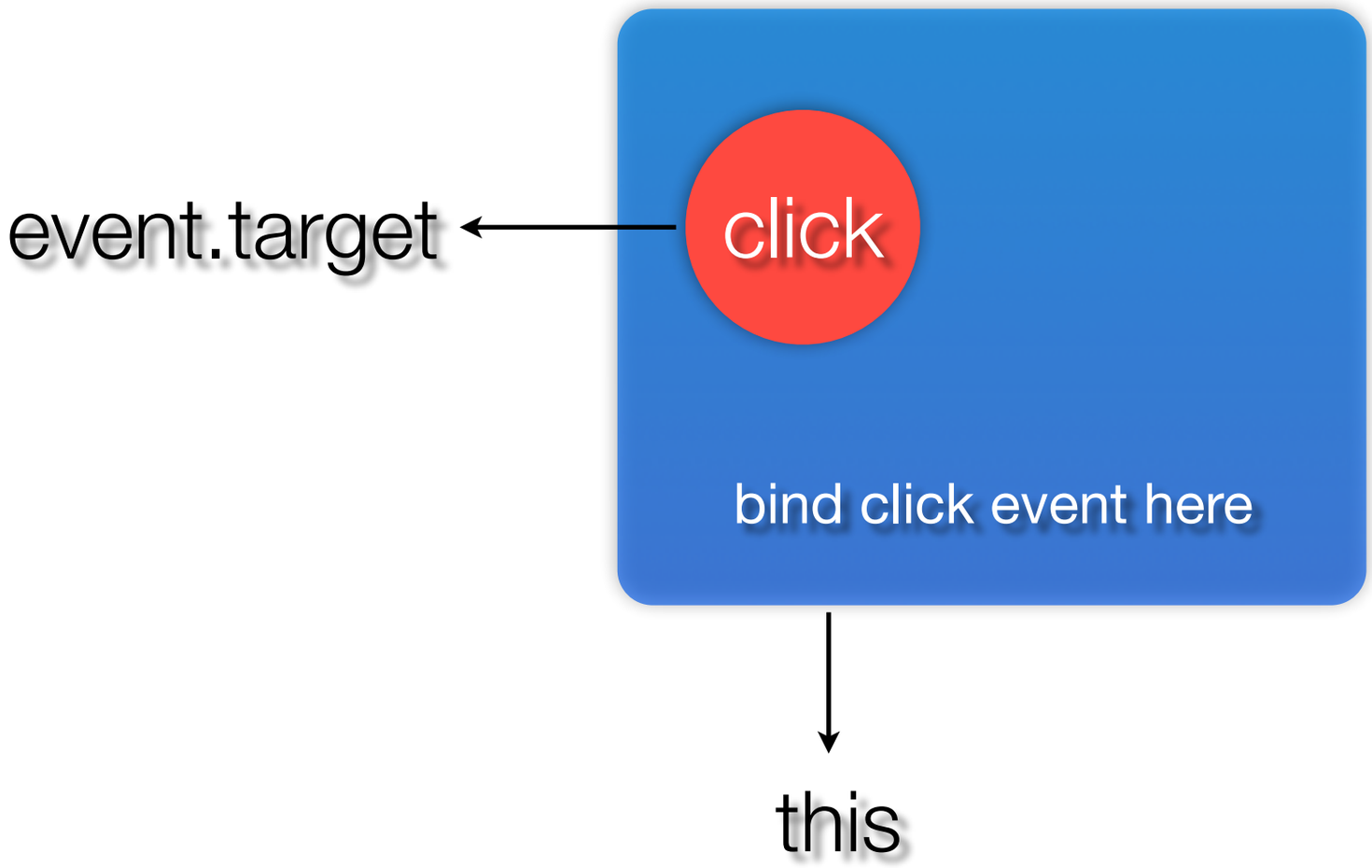
- When user use mouse to click on an element

```
node.addEventListener('click', function (event) {
  console.log('clicked');
}, false);
```

```
node.addEventListener('click', function (event) {
  console.log('clicked');
}, false);
```

Event Object

- Real target
- Mouse position
- Mouse button
- Pressed key code


event.target
click
bind click event here
this

Dev Tool Debug

Flow

- User action on document
- Trigger native events
- Trigger event listeners
- Event listener do something

Use Cases

- User mouseover item to open menu
- User click submit to send form
- User input number then calculate price

Demo

Events

click	mouseover	mouseenter	mouseleave
keydown	keyup	keypress	change
input	load	ready	unload
mousedown	mouseup	touchstart	touchcancel
touchend	play	pause	error

DOM Issues

- Hard to deep into target
- White space

XMLHttpRequest

XHR

- XMLHttpRequest
- by Microsoft, 1999
- Make HTTP request in web page

```
var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);
oReq.open("GET", "http://www.example.org/example.txt");
oReq.send();
```

```
var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);
oReq.open("POST", "http://www.example.org/example.txt");
```

```
var data = queryParams({
  foo: 1,
  bar: 2
});

oReq.send(data);
```

```
function queryParams(source) {
  var array = [];

  for (var key in source) {
    array.push(encodeURIComponent(key) + "="
      + encodeURIComponent(source[key]));
  }

  return array.join("&");
}
```

http://stackoverflow.com/questions/22582795/jquery-param-alternative-for-javascript

Form URLEncoded

```
foo=1&bar=2
```

XHR Issues

- Very long name
- API not friendly
- Need parse data

jQuery

jQuery

- 2006
- by John Resig
- over 90% market share

Technologies

Content Management
Server-side Languages
Client-side Languages
JavaScript Libraries
Markup Languages
Character Encodings
Image File Formats
Site Elements
SSL Certificate Authorities
Social Widgets
Web Servers
Operating Systems
Content Delivery
Traffic Analysis Tools
Advertising Networks
Tag Managers
Top Level Domains
Content Languages

Trends

Usage Trend
Market Share Trend

Market

Top Site Usage
Market Position

Breakdown

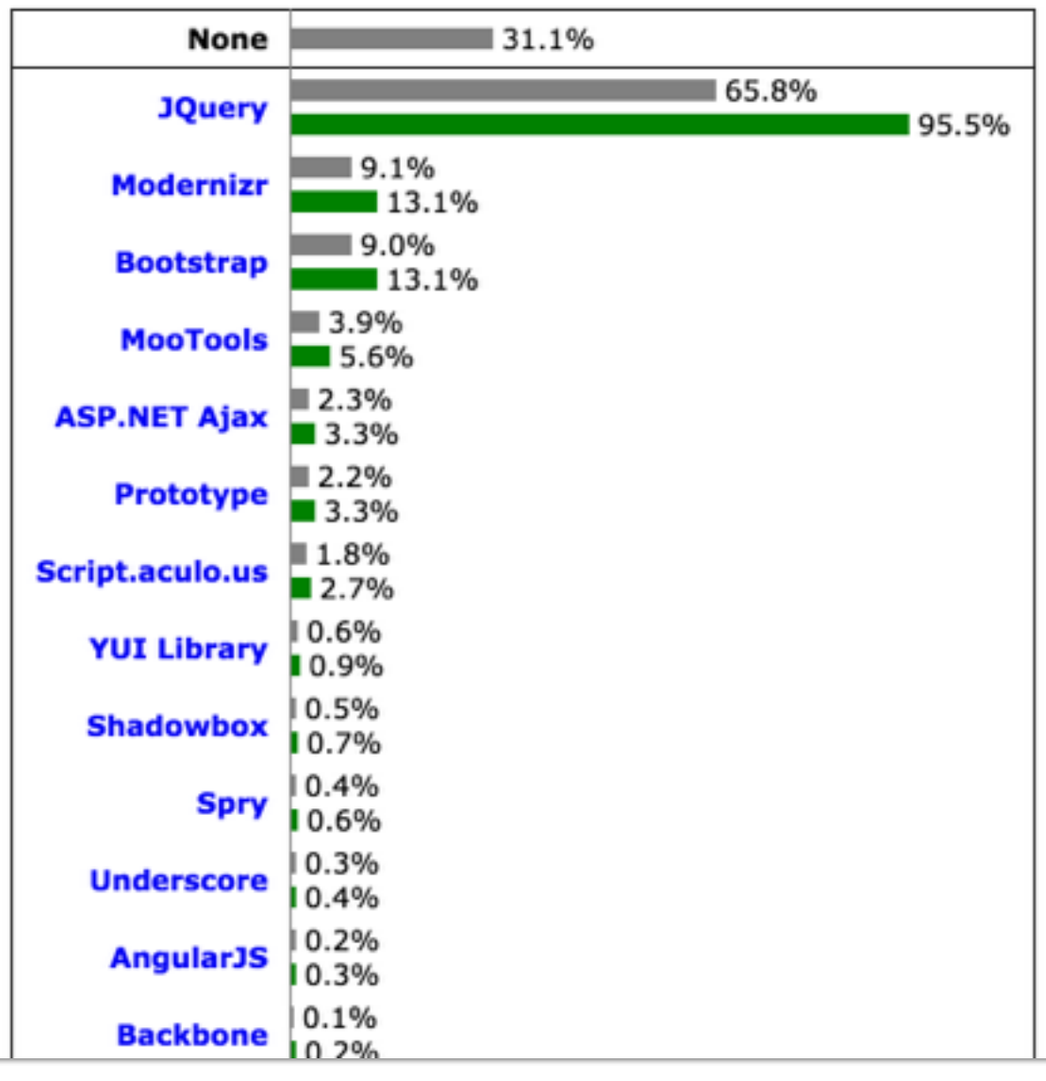
Ranking
Content Management
Server-side Languages
Client-side Languages
Markup Languages
Character Encodings
Image File Formats
Site Elements
SSL Certificate Authorities
Social Widgets

Technologies > JavaScript Libraries

Usage of JavaScript libraries for websites

This diagram shows the percentages of websites using various JavaScript libraries. See technologies overview for explanations on the methodologies used in the surveys. Our reports are updated daily.

How to read the diagram:
31.1% of the websites use none of the JavaScript libraries that we monitor.
JQuery is used by 65.8% of all the websites, that is a JavaScript library market share of 95.5%.



Technology Brief

JavaScript Libraries

JavaScript libraries provide a basis for building cross-browser web applications based on JavaScript.

Latest related posting read all

jQuery

- Solve cross browser DOM issues
- CSS selector engine
- Easy to mod DOM tree
- Event handler
- Easy to make request
- Promise

DOM Traversing

```
$('.target')
```

Append Child

```
$('.target').append(newNode);
```

Edit Attribute

```
$('.target'.target’).attr('name', 'val');
```

Chain Behavior

```
$('.target'.target’).attr('name', 'val').empty().append(newNode);
```

jQuery DEMO

Idea

- Simple naming
- Easy to use API
- Chainable operations

Effects to Standards

- querySelector API
- Promise

querySelector

```
document; // root node

document.children; // node list

document.getElementById("target"); // node #target

document.getElementsByClassName("target"); // node list .target

document.getElementsByTagName("article"); // node list article

document.querySelectorAll("article"); // node list
document.querySelectorAll(".target, #target"); // node list
```

```
document; // root node

document.children; // node list

document.getElementById("target"); // node #target

document.getElementsByClassName("target"); // node list .target

document.getElementsByTagName("article"); // node list article

document.querySelectorAll("article"); // node list
document.querySelectorAll(".target, #target"); // node list

document.querySelector("#target"); // node
```

Promise

- A future value
- ex: HTTP request response
- Use callback in jQuery

http://www.flickr.com/photos/gozalewis/3256814461/

jQuery XHR DEMO

JavaScript Patterns

shichuan.github.io/javascript-patterns/

Fork me on GitHub

JavaScript Patterns Collection

ShareThis 102K | Tweet 7973 | Share 5139 | G+1 1,536

- General Patterns
- jQuery Patterns
- jQuery Plugin Patterns
- Literals and Constructor Patterns
- Function Patterns
- Object Creation Patterns
- Code Reuse Patterns
- Design Patterns

A JavaScript pattern and antipattern collection that covers function patterns, jQuery patterns, jQuery plugin patterns, design patterns, general patterns, literals and constructor patterns, object creation patterns, code reuse patterns, DOM and browser patterns (*upcoming*).

Patterns collected while developing 喜感网.

General Patterns

Function Declarations - creating anonymous functions and assigning them to a variable

Conditionals - pattern and antipattern of using if else

Access to the Global Object - access the global object without hard-coding the identifier window

Single var Pattern - use one var statement and declare multiple variables

Hoisting - var statements anywhere in a function act as if the variables were declared at the top of the function

for loops - optimized for loops

for-in loops - optimized for-in loops

(Not) Augmenting Built-in Prototypes - only augment built-in prototypes when certain conditions are met

switch Pattern - improve the readability and robustness of your switch statements

Implied Typecasting - avoid implied typecasting

Avoiding eval() - avoid using eval()

more BOM

new Web APIs

- localStorage
- IndexedDB
- Fullscreen
- Notification
- Geolocation
- Orientation
- Page Visibility
- …

- https://developer.mozilla.org/en-US/docs/Web/API
- https://wiki.mozilla.org/WebAPI

JavaScript 就要统治世界了？ ×

segmentfault.com/a/1190000003767058

segmentfault 输入关键字搜索 问答 文章 活动 榜单 ··· 注册 · 登录

文章 > PuYart > 文章详情

JavaScript 就要统治世界了?

PuYart 631 3 天前 发布

推荐 18 推荐

收藏 91 收藏，6.8k 浏览

> " JavaScript 可以……"
> "嘛，不就是操作一下 DOM，可以让元素飞来飞去吗"
> "JavaScript 是……"
> "不就是用 jQuery 让网页动起来，顶多就是再用用 Ajax 和后端进行一下数据交换吗"
> "JavaScript 是一门……"
> "最讨厌和鄙视这种弱类型不需要编译的脚本语言了，你说 OOP? 扯淡的吧，JS 有对象吗"
> "……"

本文隶属于专栏

PuYart

PuYart 扯技术，胡说话的地方

关注专栏

文章目录

0x00. 前言

早上起床惯例刷刷微博，突然看到 React Native 宣布支持 Android 的消息，一时感觉 Facebook 太给力了，不仅没有推迟发布 React Native For Android 而且还比之前公告的时间提前了一些。立马下床打开电脑赶紧上官网，心想着用 JS 写原生安卓的日子终于要来了。乐乐呵呵地打开文档，然后瞬间就傻眼了。好吧，尽欺负我们这些买不起 Mac 的穷学生。

Requirements

1. OS X - Only OS X is currently supported
2. Homebrew is the recommended way to install nvm, watchman, and flow.
3. Install Node.js 4.0 or newer.
   - Install **nvm** with Homebrew or its setup instructions here. Then run `nvm install node && nvm alias default node`, which installs the latest version of Node.js and sets up your terminal so you can run it by typing `node`. With nvm you can install multiple versions of Node.js and easily switch between them.
   - New to npm?
4. `brew install watchman`. We recommend installing watchman, otherwise you might hit a