



# Lecture 09. Web Servers & Server Side JavaScript

**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

# Outline

---

- **Web Servers**
- Node.js in a glance
- Form

# Web is a Distributing / Sharing System

拼图游戏

Steps: 0

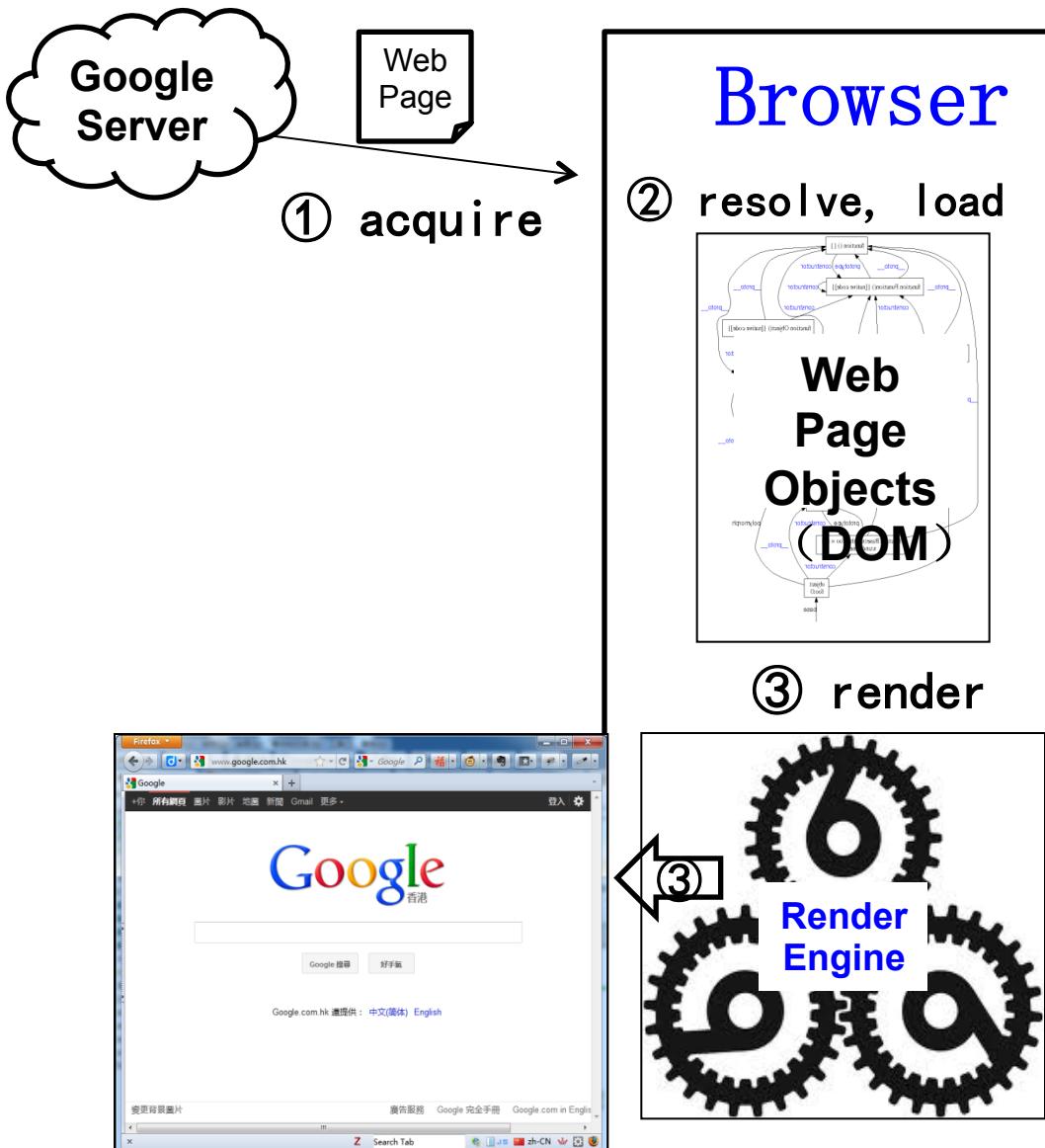


Disney PIRATES OF THE CARIBBEAN  
— AT WORLD'S END —

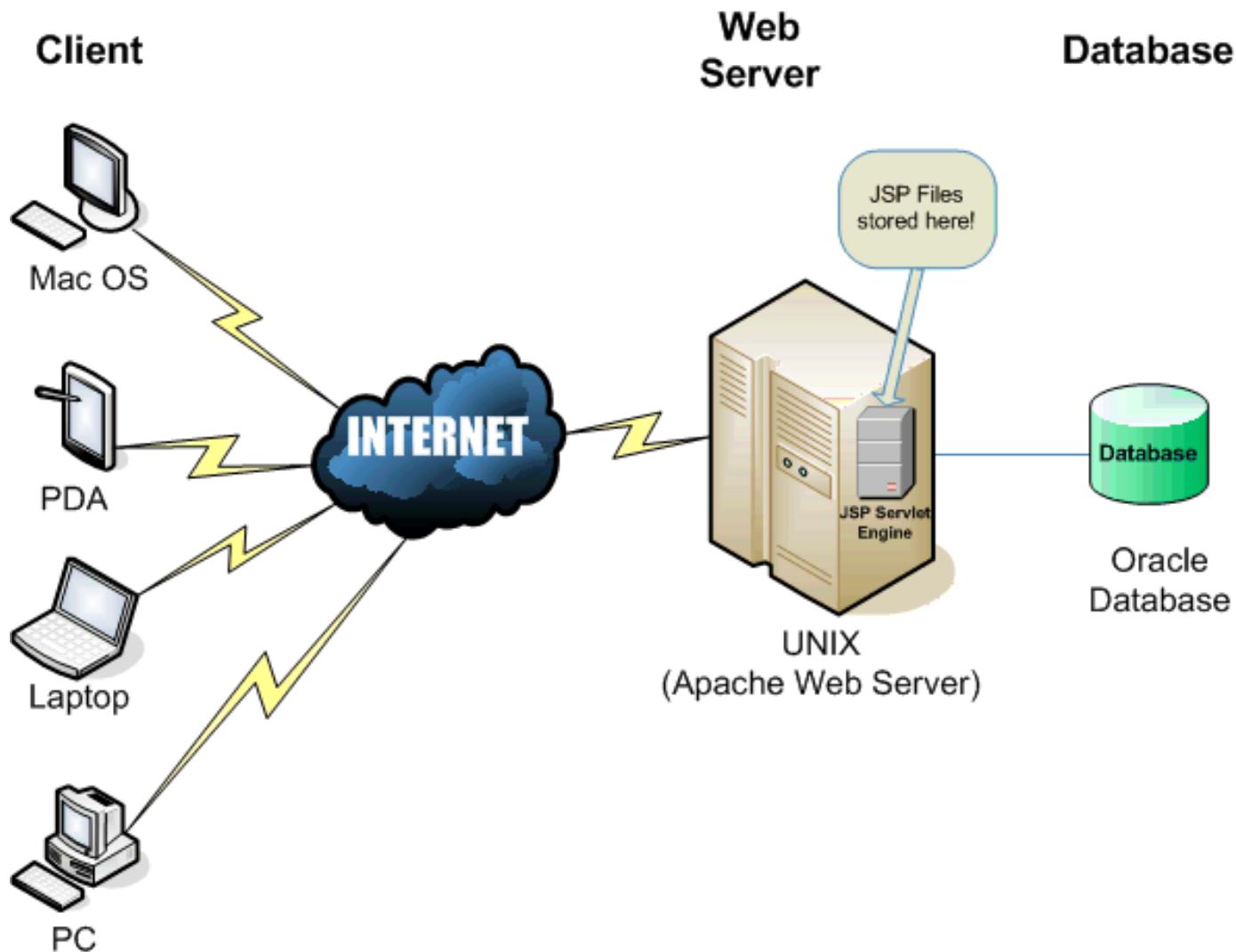
How can I share web pages on the Web?

RESTORE

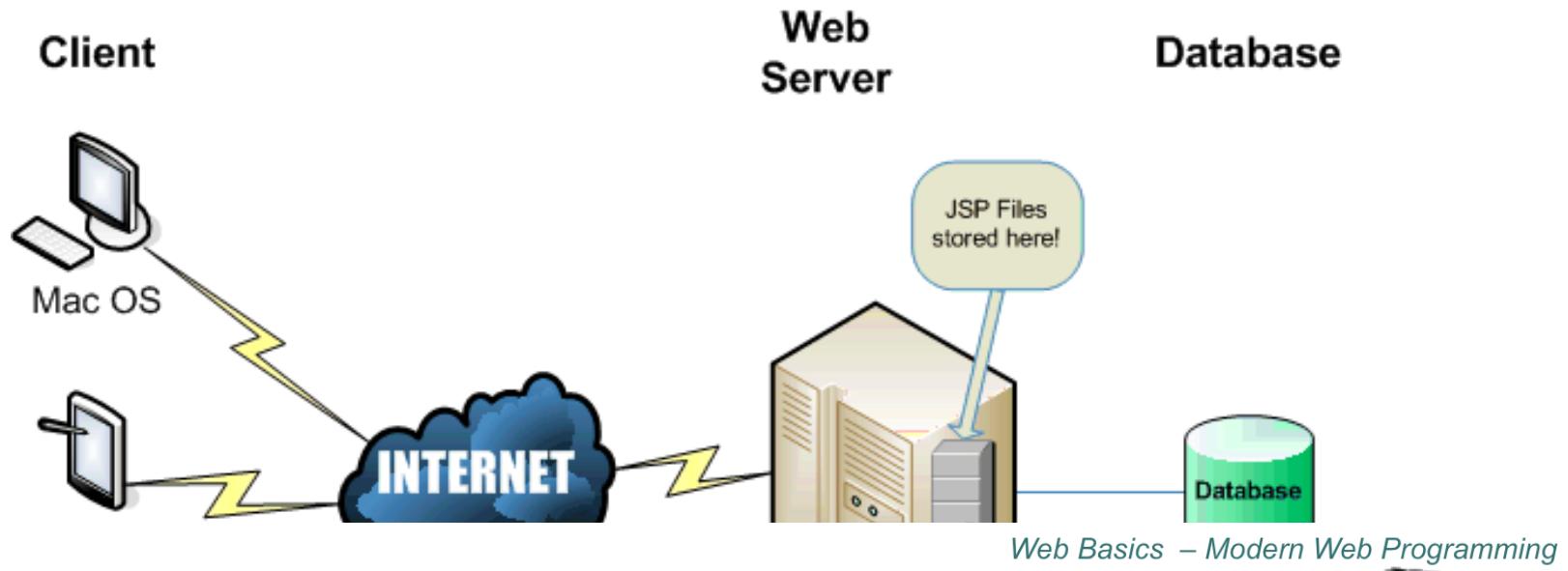
# Working mechanism of browsers



# Web is a Distributing / Sharing System



# Web is a Distributing / Sharing System



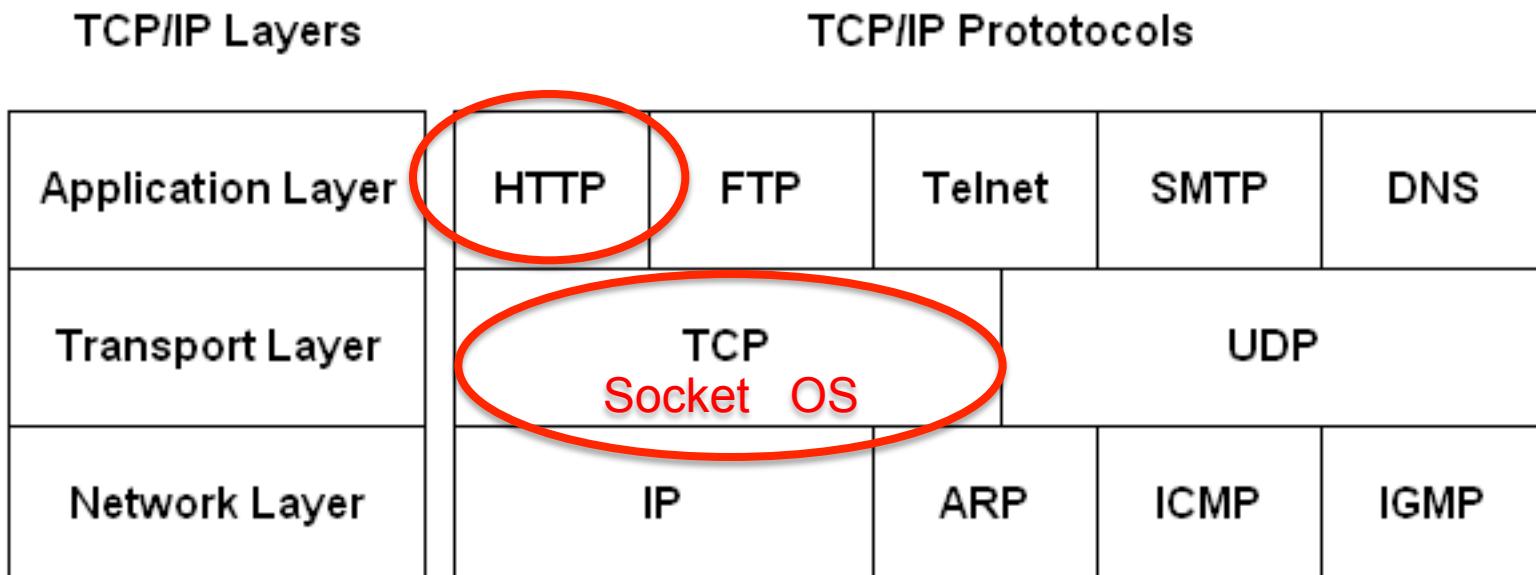
## Web servers and browsers

- **Web server:** software that listens for Web page requests
  - Apache
  - Microsoft Internet Information Server (IIS)  
(part of Windows)



# Web server

- Web = HTML\* + HTTP(S)



Web server is a software that computes HTML\* content and then delivers them via HTTP(s) to browsers

# Web Servers

---



VARNISH CACHE



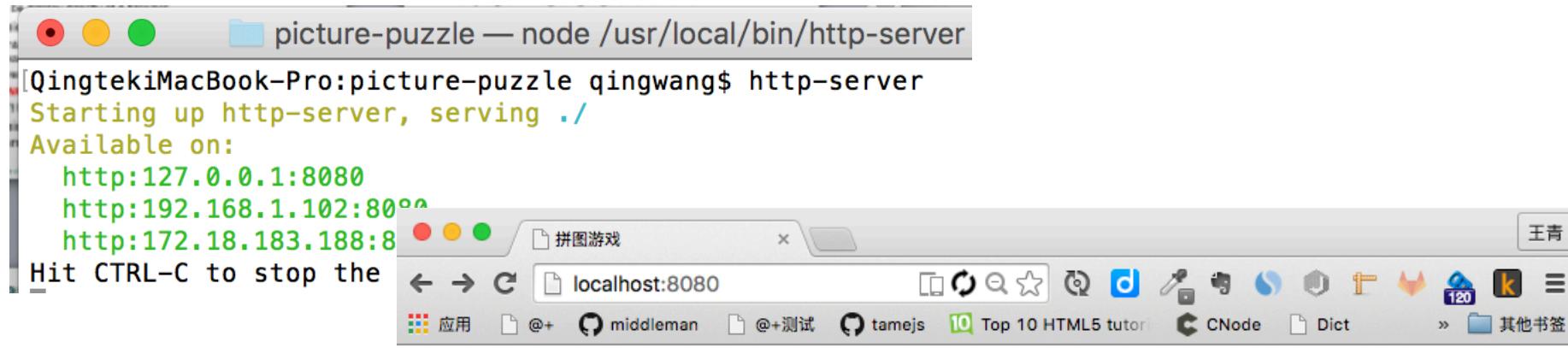
cherokee

NGINX

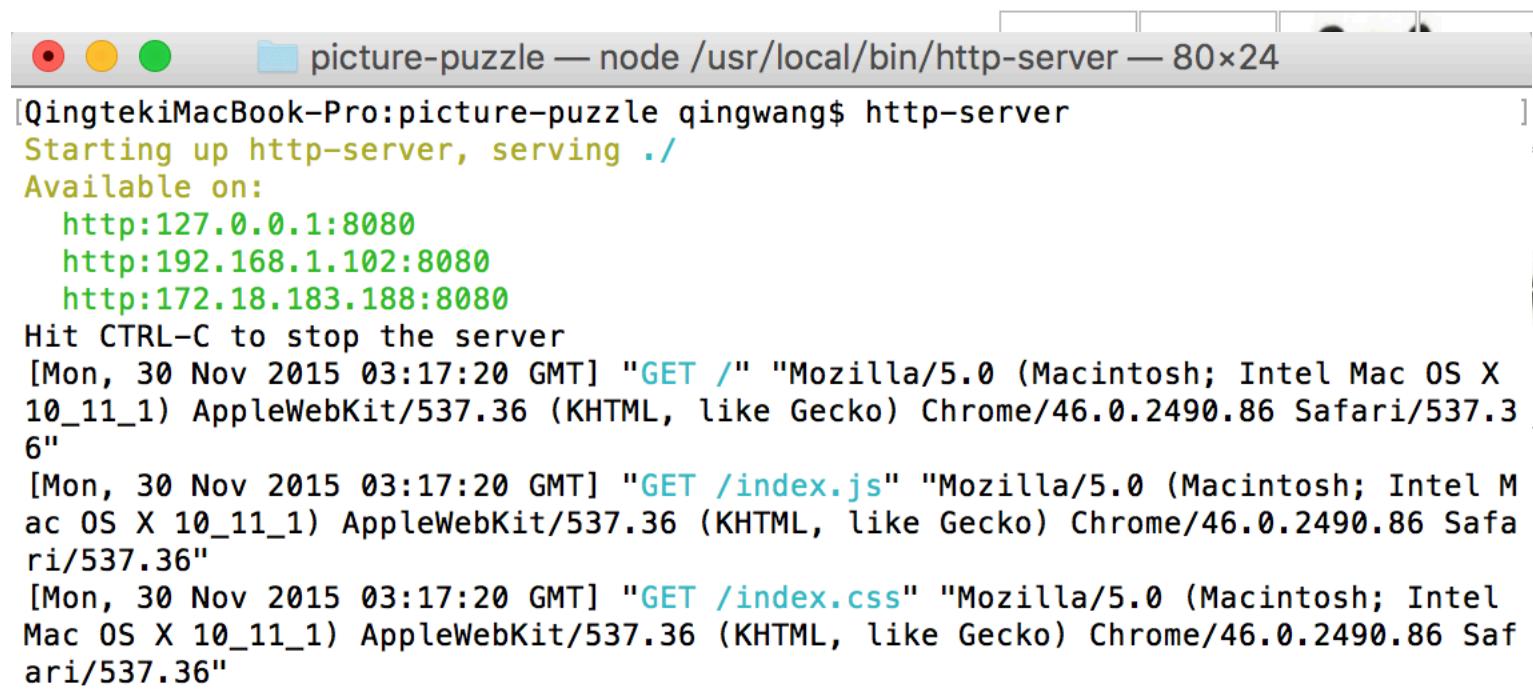


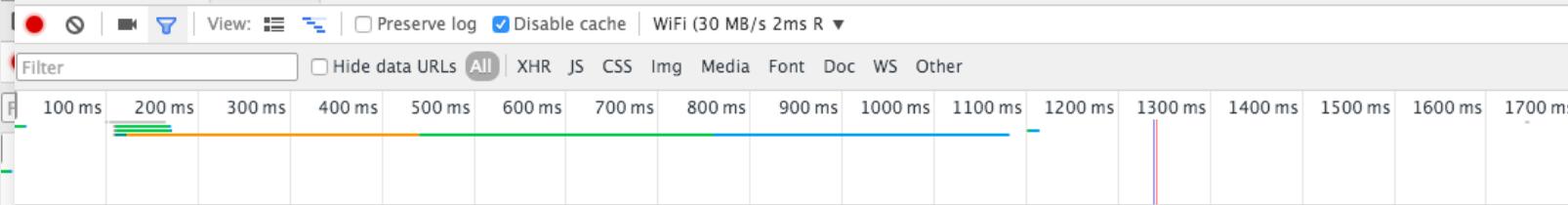
G-WAN web server

# Share Fifteen Puzzle



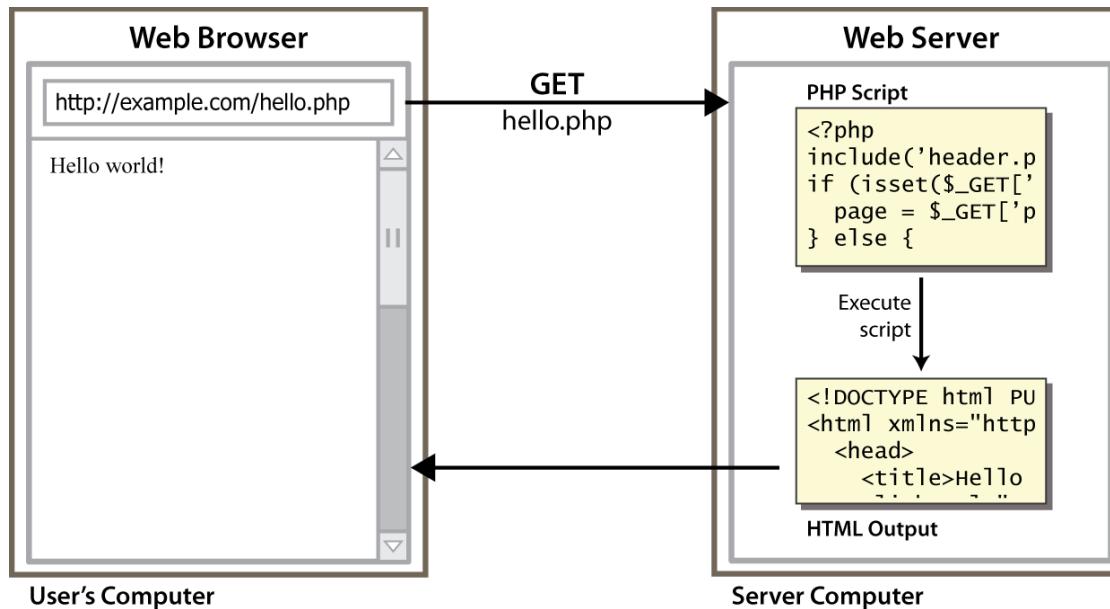
## 拼图游戏





Name	Headers	Preview	Response	Timing
localhost			<pre>1 &lt;!DOCTYPE html&gt; 2 &lt;html&gt; 3   &lt;head&gt; 4     &lt;meta charset="utf-8"&gt; 5     &lt;meta http-equiv="X-UA-Compatible" content="IE=edge"&gt; 6     &lt;title&gt;拼图游戏&lt;/title&gt; 7     &lt;script type="text/javascript" src='http://code.jquery.com/jquery-2.1.4.min.js'&gt;&lt;/script&gt; 8     &lt;script type="text/javascript" src='index.js'&gt;&lt;/script&gt; 9     &lt;link rel="stylesheet" href="index.css"&gt; 10   &lt;/head&gt; 11   &lt;body&gt; 12     &lt;div id='container'&gt; 13       &lt;h1&gt;拼图游戏&lt;/h1&gt; 14       &lt;div id='puzzle-pane'&gt; 15         &lt;div id='1'&gt;&lt;/div&gt; 16         &lt;div id='2'&gt;&lt;/div&gt; 17         &lt;div id='3'&gt;&lt;/div&gt; 18         &lt;div id='4'&gt;&lt;/div&gt; 19         &lt;div id='5'&gt;&lt;/div&gt; 20         &lt;div id='6'&gt;&lt;/div&gt; 21         &lt;div id='7'&gt;&lt;/div&gt; 22         &lt;div id='8'&gt;&lt;/div&gt; 23         &lt;div id='9'&gt;&lt;/div&gt; 24         &lt;div id='10'&gt;&lt;/div&gt; 25         &lt;div id='11'&gt;&lt;/div&gt; 26         &lt;div id='12'&gt;&lt;/div&gt; 27         &lt;div id='13'&gt;&lt;/div&gt; 28         &lt;div id='14'&gt;&lt;/div&gt; 29         &lt;div id='15'&gt;&lt;/div&gt; 30       &lt;/div&gt; 31       &lt;button&gt;重新开始&lt;/button&gt; 32     &lt;/div&gt; 33   &lt;/body&gt; 34 &lt;/html&gt;</pre>	

# Lifecycle of Dynamic Web request



- browser requests a .html file (**static content**): server just sends that file
- browser requests **dynamic content**: server runs code, then sends result across the network
  - script produces output that becomes the response sent back

# Dynamic Vs. Static

---

- Static Page
  - Client/Consumer's Viewpoint: an url refers to an identical html file
  - Server/Producer's Viewpoint: a file stored within or sub-within the root folder of a Web Server
  - it is a html ...
  - Can be display directly at a browser
- Dynamic Page
  - Client/Consumer's Viewpoint: an url refers to a dynamic html (may be vary each time requested)
  - Server/Producer's Viewpoint: a program/script produces html
  - it is **NOT a html**, but a program producing html(s)
  - Can't be display directly at a browser
- Dynamic Web Page, Dynamic HTML (DHTML), what's the difference?

# Server-Side Web Programming



- server-side pages are programs written using one of many web programming languages/frameworks
  - examples: [PHP](#), [Java/JSP](#), [Ruby on Rails](#), [ASP.NET](#), [Python](#), [Perl](#)
- the web server contains module that executes those programs and send back their output as responses to web requests
- each language/framework has its pros and cons
- We use JavaScript!

# Outline

---

- Web Servers
- **Node.js in a glance**
- Form





[HOME](#) | [ABOUT](#) | [DOWNLOADS](#) | [DOCS](#) | [FOUNDATION](#) | [GET INVOLVED](#) | [SECURITY](#) | [NEWS](#)

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, [npm](#), is the largest ecosystem of open source libraries in the world.

## Download for OS X (x64)

**v4.2.2 LTS**

Mature and Dependable

**v5.1.0 Stable**

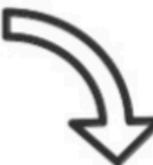
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

<https://nodejs.org>

Google



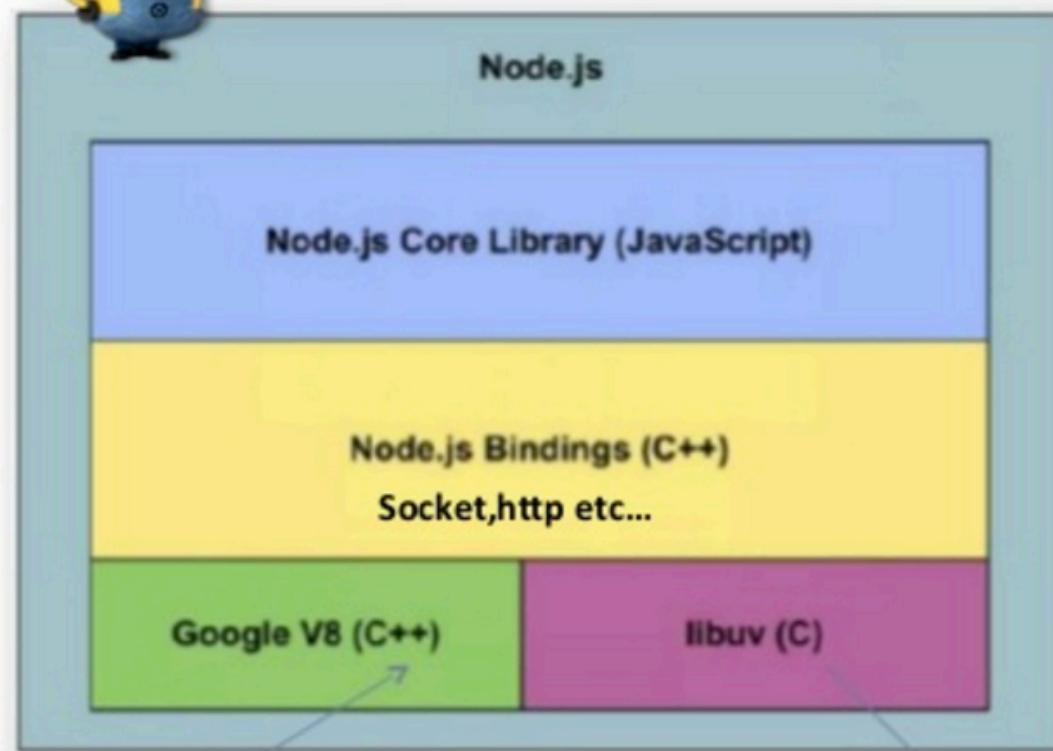
Gmail  
by Google



# NodeJS under the hood...



Callbacks + Polling  
Epoll, POSIX AIO, Kqueue ...



Open source JavaScript engine, platform Optimizer, JIT, inline caching, GC

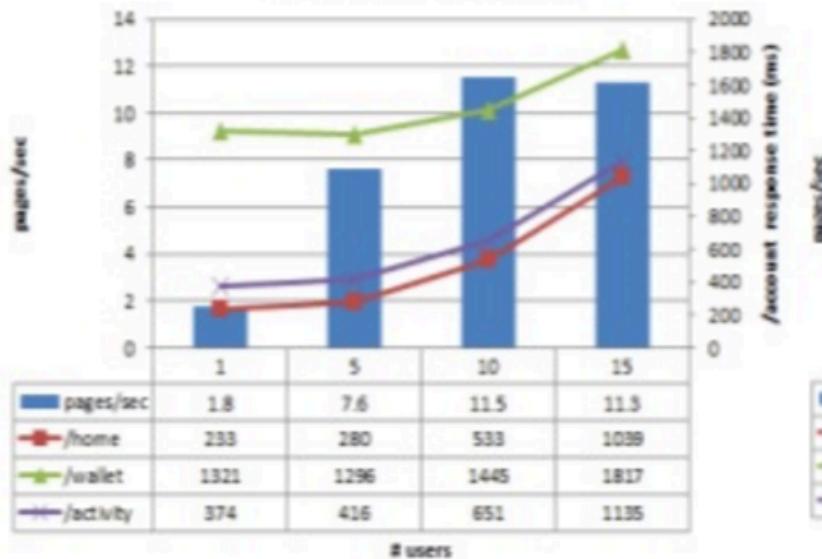
Cross-platform asynchronous I/O. Event Loop, Worker Thread Pool ,



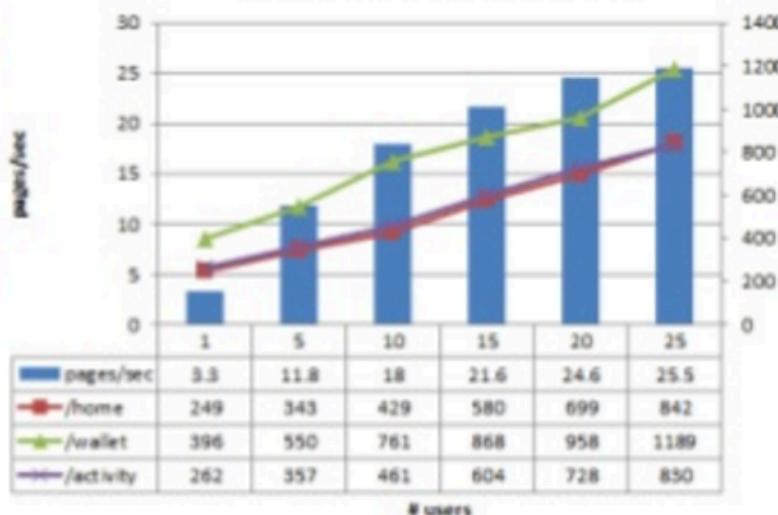
NodeJS ecosystem - Yukti Kaura

# The PAYPAL STORY

Java application



Node.js application



- Double the requests per second vs. the Java application.
- 35% decrease in the average response time for the same page. This resulted in the pages being served 200ms faster— something users will definitely notice.



# THE **PayPal** BENEFITS

---



1. **Full-stack engineers.** Using JavaScript on both the front-end and the back-end removed an artificial boundary between the browser and server, allowing engineers to code both.
2. Built almost **twice as fast with fewer people**
3. Written in **33% fewer lines of code**
4. Constructed with **40% fewer files**
5. **Double the requests per second** vs. the Java application.
6. **35% decrease in the average response** time for the same page



An example of a **web server** written with Node.js which responds with 'Hello World':

```
var http = require('http');

http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(8124);

console.log('Server running at http://127.0.0.1:8124/');
```

To run the server, put the code into a file called `example.js` and execute it with the node program

```
> node example.js
Server running at http://127.0.0.1:8124/
```

# Node.js (1)

About these Docs

## Synopsis

Assertion Testing

Buffer

C/C++ Addons

Child Processes

Cluster

Console

Crypto

Debugger

DNS

Domain

Errors

Events

File System

Globals

HTTP

HTTPS

Modules

Net

OS

Path

Process

Punycode

Query Strings

Readline

REPL

Stream

String Decoder

Timers

TLS/SSL

TTY

UDP/Datagram

URL

Utilities

V8

VM

ZLIB

<https://nodejs.org/api/>

# Hello somebody

---

```
1 var http = require('http');
2 var url = require('url');
3 var querystring = require('querystring')
4
5 function parseName(_url){
6   return querystring.parse(url.parse(_url).query).username;
7 }
8
9 http.createServer(function (request, response) {
10   response.writeHead(200, {'Content-Type': 'text/plain'});
11   response.end('Hello ' + (parseName(request.url) || 'world') + '!\\n');
12 }).listen(8124);
13
14 console.log('Server running at http://127.0.0.1:8124/');
```



Don't reinvent the wheel





find packages



sign up or log in



# npm is the package manager for javascript.

**210,597**

total packages

**41,046,004**

downloads in the last day

**649,959,955**

downloads in the last week

**2,708,391,143**

downloads in the last month

## packages people 'npm install' a lot

**browserify**

browser-side require() the node way

10.2.6 published 5 months ago by substack

**grunt-cli**

The grunt command line interface.

0.1.13 published 2 years ago by tkellen

**bower**

The browser package manager

1.4.1 published 8 months ago by sheerun

**gulp****express** **express**

Fast, unopinionated, minimalist web fra...

4.13.1 published 5 months ago by dougwilson

**npm**

a package manager for JavaScript

2.13.0 published 5 months ago by zkat

**cordova**

Cordova command line interface tool

5.1.1 published 6 months ago by stevegill

**forever****pm2**

Production process manager for Node.JS...

0.14.3 published 5 months ago by jshkurti

**karma**

Spectacular Test Runner for JavaScript.

0.13.1 published 4 months ago by dignifiedq...

**coffee-script**

Unfancy JavaScript

1.9.3 published 6 months ago by jashkenas

**statsd**

## Installing and updating npm

**Npm comes with the installation of Node.js.**

**To update npm simple type**

```
npm install npm -g
```

# package.json

Where our  
packages  
are installed

What  
version of  
the package  
do we need

```
{  
  "name": "example-app",  
  "version": "1.0.0",  
  "dependencies": {  
    "underscore": "^2.0.0",  
    "jQuery": "*"  
  }  
}
```

# List of all npm commands

- access
- adduser
- bin
- bugs
- build
- bundle
- cache
- completion
- config
- dedupe
- deprecate
- dist-tag
- docs
- edit
- explore
- help
- help-search
- init
- install
- link
- logout
- ls
- npm
- outdated
- owner
- pack
- ping
- prefix
- prune
- publish
- rebuild
- repo
- restart
- rm
- root
- run-script
- search
- shrinkwrap
- star
- stars
- start
- stop
- tag
- test
- uninstall
- unpublish
- update
- version
- view
- whoami



Dont panic you don't need to use around 70%!

# Outline

---

- Web Servers
- Node.js in a glance
- **Form**

# Dynamic/Parameterized Web page

查询字符串（Query Strings）是浏览器向服务器提交参数的基本手段之一。查询字符串通过问号“?”附加在 URL 末，由以“&”分隔的多个参数名值对构成，参数名和参数值用等号“=”连接，参考源代码 6-1。

## 源代码 6-1 有查询字符串的 URL 基本模式

```
URL?paramName1=value1&paraName2=value2&...
```

## 源代码 6-2 有查询字符串的 URL 示例

```
http://www.google.com/search?q=Web+2.0+programming&start=10
```

- 地址栏手动输入？
- 网页作者写在 href 里？

# Form (表单)

表 6-1 表单元素 (form)

HTML 元素名	用途	语法
<u>form</u>	提供给用户录入数据，并提交到 Web 服务器(块元素)	< <u>form</u> <u>action</u> =" <u>Server URL</u> "> 表单控件 表单控件 ..... </ <u>form</u> >

表单元素有一个重要属性 `action`，该属性定义了当用户提交表单的时候，需要提交到的服务器 URL。

# Form (表单)

## 源代码 6-3 网页中加入 Google 搜索框

其它内容.....

```
<form action="http://www.google.com.hk/search?q=">
    Google:
    <input name="q" />
    <input type="submit" value="搜尋">
</form>
```

其它内容.....

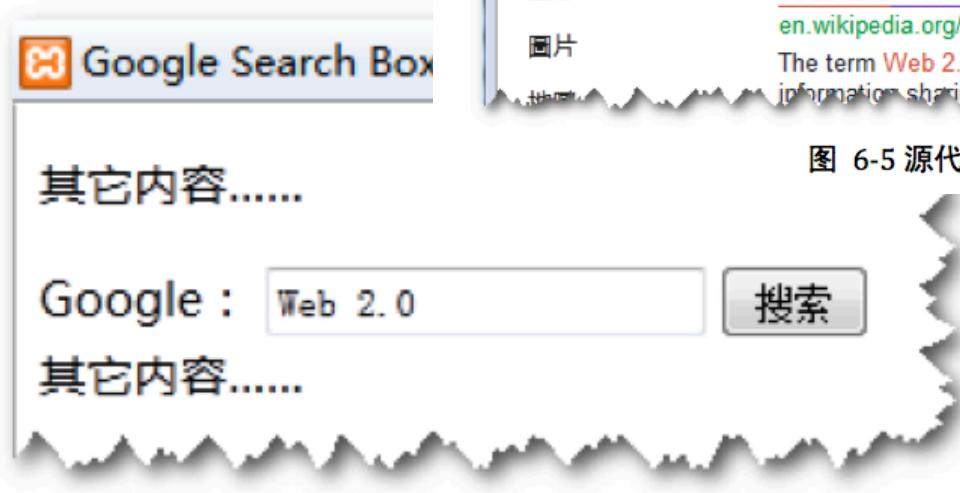


图 6-5 源代码 6-3 运行效果

# Form controls

表 6-2 表单控件元素 (input)

HTML 元素名	用途	语法
<code>input</code>	用户在网页上录入数据	<code>&lt;input type="控件类型" 其它属性/&gt;</code>

表单控件的类型有很多中，通过其 `type` 属性来区分，不同 `type` 代表了不同数据录入界面和方式的控件。`type` 的默认值是“text”，就是单行文本框，参见图 6-5。`type` 为“submit”时，表单控件为一个提交按钮，如图 6-5 中的“搜索”按钮，用户点此按钮，表单就被提交给服务器。这两个表单控件都是空元素，只有一个标签。



`name` 属性：除了提交和清除按钮（参见 6.3.7），其余表单控件均用来录入用户数据，它们都有应该有 `name` 属性。`name` 属性十分重要，它用来向服务器说明该控件的值绑定到那个参数。源代码 6-3 中的文本框 `name` 属性为“q”，说明文本框的值将会绑定到参数 q。



**value** 属性：所有的表单控件都有 **value** 属性，对于提交按钮，**value** 属性说明了出现在按钮上的文字，参见源代码 6-3。对于其它按钮，**value** 属性给出了控件的初始默认值。例如中，打开网页搜索框就会出现初始默认值“Modern Web”。

#### 源代码 6-4 搜索框初始默认值示例

其它内容.....

```
<form action="http://www.google.com/search">
    Google:
    <input name="q" value="Modern Web"/>
    <input type="submit" value="搜索"/>
</form>
```

其它内容.....

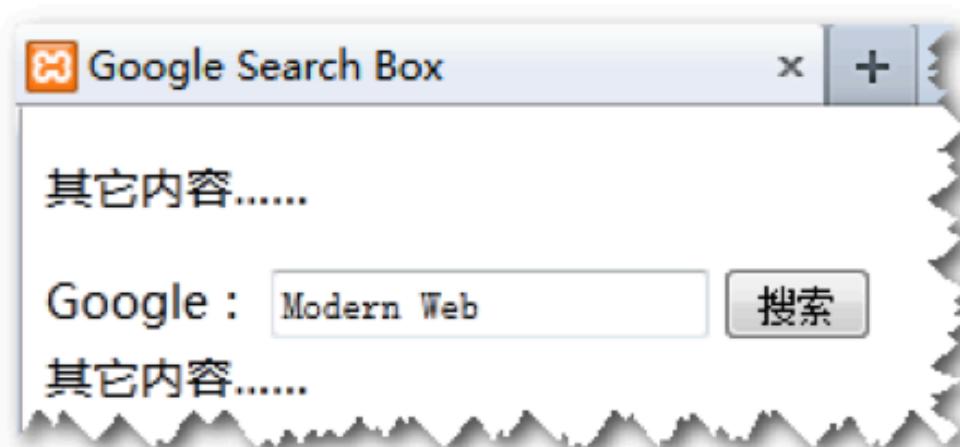
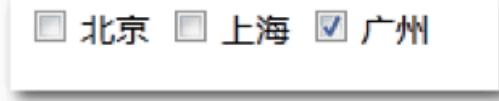


图 6-6 源代码 6-4 运行效果

表 6-3 表单控件元素外观一览

表单控件名	示例
文本框（单行） <u>type="text"</u>	
文本域（多行） <u>textarea</u>	
提交、清除按钮 <u>type="submit"</u> <u>type="reset"</u>	
多选框 <u>type="checkbox"</u>	
单选框 <u>type="radio"</u>	
下拉菜单和列表 <u>select</u>	

所有表单控件都是行内元素

必须嵌套在块元素

通常是表单（form）内使用

**`$(...).val`** 操作表单控件的值

# GET vs. POST

姓名: 张三  
密码: ●●●●  
套餐: 扬州炒饭  
加饭?   
**下单**    **清除**

method="GET"

图 6-18 订餐表单



# GET vs. POST

源代码 6-19 POST 提交表单示例

**Source Code 6-19 POST Submission Form Example:**

```
<form action="params.php" method="post">
    <label>姓名: <input type="text" name="name" /></label><br />
    <label>密码 /></label><br />
    <label>套餐 /></label><br />
    <label>加饭 />
    <input type="checkbox" name="meal" value="扬州炒饭" checked="checked" />
    <input type="checkbox" name="meal" value="宫保鸡丁" />
</form>
```

The screenshot shows a browser window with the URL `localhost/eclipse/MWP5/params.php`. Below the browser is a Firebug interface. The 'Post' tab of the Network panel shows the following data:

URL	Status
POST params.php	200 OK

The 'Parameters' section shows the POST data:

Parameters	Value
add	on
meal	扬州炒饭
name	张三
password	1234

图 6-20 POST 方法提交后的结果网页和 Firebug 显示的 HTTP 报文

# GET vs. POST

除了“简单查询，不改变服务器状态使用 **GET**；否则，使用 **POST**”这条原则，在使用 GET 和 POST 方法提交数据时，还有一些特点值得注意。



**URL 长度限制：**虽然 [URL 标准<sup>4</sup>](#)并未定义 URL 长度限制，但是各浏览器在实现中却有不同的 [URL 长度限制<sup>5</sup>](#)。例如，IE 的 URL 长度不超过 2083 个字符。所以，当提交的参数和数据较多的时候，不能使用 GET 方法，否则会被浏览器截断。



**重复提交：**按照原则，GET 方法不改变服务器状态，因此重复提交不会带来问题，而 **POST** 方法会。图 6-21 展示了一个 Firefox 浏览器显示的“Confirm”对话框，提醒用户“为了显示此页，Firefox 必须发送信息，从而重复任何操作（如搜索或订单确认），这是之前执行的操作。”

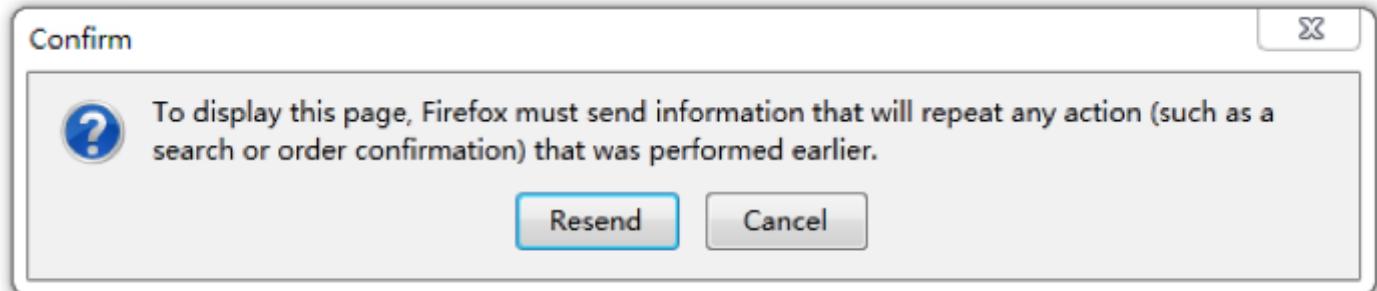


图 6-21 重复提交警示对话框

# What is form validation?

---

- **validation:** ensuring that form's values are correct
- some types of validation:
  - preventing blank values (email address)
  - ensuring the type of values
    - integer, real number, currency, phone number, Social Security number, postal address, email address, date, credit card number, ...
  - ensuring the format and range of values (ZIP code must be a 5-digit integer)
  - ensuring that values fit together (user types email twice, and the two must match)

# A real form that uses validation



**WaMu**

[Cancel](#)

[Secure Site](#)

Questions? Call us:  
**(800) 788-7000**

## Signing Up is Easy

**⚠ Some of the information you entered is missing or incorrect. Please check all highlighted messages below.**

- ⚠ Please enter Last Name using letters, apostrophes or dashes.**
- ⚠ Enter a valid date for Date of Birth.**
- ⚠ Please enter a valid e-mail address.**

### Personal Info

First Name:

Last Name:

Date of Birth:   Day  Year

E-mail Address:

Identify yourself by your:

- Account Number
- ATM/Debit Card
- Credit Card

Step 1

**CONTINUE ➔**

# Client vs. serve-side validation

---

- Validation can be performed:
- **client-side** (before the form is submitted)
  - can lead to a better user experience, but not secure (why not?)
- **server-side** (in server code, after the form is submitted)
  - needed for truly secure validation, but slower
- **both**
  - best mix of convenience and security, but requires most effort to program

# RegEx

Regular Expression

/h[a4@](([c<](k)|(\\<))|((k)|(\\<))(x))\\s+\\  
((d)|(t\\+)[h])[3ea4@]\\s+p[l1][a4@]n[3e][t\\+]/i

(C)2006 FTS Conventions - [www.ftscconventions.com](http://www.ftscconventions.com)

# What is a regular expression?

```
"/^ [a-zA-Z_-]+@[([a-zA-Z_-]+)\.][a-zA-Z]{2,4}$/"
```

- **regular expression ("regex")**: a description of a pattern of text
  - can test whether a string matches the expression's pattern
  - can use a regex to search/replace characters in a string
- regular expressions are extremely powerful but tough to read
- (the above regular expression matches email addresses)
- regular expressions occur in many places:
  - Java: **Scanner**, String's **split** method
  - supported by PHP, JavaScript, and other languages
  - many text editors (**Notepad++**, **TextPad**) allow regexes in search /replace

# Basic regular expressions

---

```
> /abc/.constructor
```

```
< function RegExp() { [native code] }
```

- in JavaScript, regexes are literal that begin and end with /
- the simplest regexes simply match a particular substring
- the above regular expression matches any string containing "abc":
  - YES: "abc", "abcdef", "defabc", ".=.abc.=.", ...
  - NO: "fedcba", "ab c", "JavaScript", ...

```
> /abc/.test('my abc is good')
```

```
< true
```

```
> 'my abc is good'.match(/abc/)
```

```
< ["abc"]
```

```
> 'my abc is good'.match(/abe/)
```

```
< null
```

# Wildcards: .

---

- A dot `.` matches any character except a `\n` line break
  - `"/.oo.y/"` matches "Doocy", "goofy", "LooNy", ...
- A trailing `i` at the end of a regex (after the closing `/`) signifies a case-insensitive match
  - `"/mart/i"` matches "Marty Stepp", "smart fellow", "WALMART", ...

# Special characters: |, (), ^, \

- | means OR
  - `"/abc|def|g/"` matches "abc", "def", or "g"
  - There's no AND symbol. Why not?
- () are for grouping
  - `"/(Homer|Marge) Simpson/"` matches "Homer Simpson" or "Marge Simpson"
- ^ matches the beginning of a line; \$ the end
  - `"/^<!--$/"` matches a line that consists entirely of "<!--"
- \ starts an escape sequence
  - many characters must be escaped to match them literally: / \ \$ . [ ] ( ) ^ \* + ?
  - `"/<br \\\r>/"` matches lines containing `<br />` tags

# Quantifiers: \*, +, ?

---

- \* means 0 or more occurrences
  - `/abc*/` matches "ab", "abc", "abcc", "abccc", ...
  - `/a(bc)*/` matches "a", "abc", "abcbc", "abcbcbc", ...
  - `/a.*a/` matches "aa", "aba", "a8qa", "a!?!\_a", ...
- + means 1 or more occurrences
  - `/a(bc)+/` matches "abc", "abcbc", "abcbcbc", ...
  - `/Goo+gle/` matches "Google", "Google", "Goooogle", ...
- ? means 0 or 1 occurrences
  - `/a(bc)?/` matches "a" or "abc"

# More quantifiers: {min,max}

- **{min,max}** means between *min* and *max* occurrences (**inclusive**)
  - `"/a(bc){2,4}/"` matches "abcbc", "abcbcbc", or "abcbcbcbc"
- *min* or *max* may be omitted to specify any number
  - `{2,}` means 2 or more
  - `{,6}` means up to 6
  - `{3}` means exactly 3

# Character sets: []

---

- [] group characters into a character set; will match any single character from the set
  - `"/[bcd]art/"` matches strings containing "bart", "cart", and "dart"
  - equivalent to `"/(b|c|d)art/"` but shorter
- inside [], many of the modifier keys act as normal characters
  - `"/what[!*?]*/"` matches "what", "what!", "what?\*\*!", "what??!", ...
- What regular expression matches DNA (strings of A, C, G, or T)?

# Character ranges: [start-end]

---

- inside a character set, specify a range of characters with -  
  `"/[a-z]/"` matches any lowercase letter
- `"/[a-zA-Z0-9]/"` matches any lower- or uppercase letter or digit
- an initial `^` inside a character set **negates** it `"/[^abcd]/"`  
  matches any character other than a, b, c, or d
- inside a character set, `-` must be escaped to be matched  
  `"/[+\-]?[0-9]+/"` matches an optional + or -, followed by at least one digit
- What regular expression matches letter grades such as A, B+, or D- ?  
  `"/[ABCDF][+\-]?/"`

# Escape sequences

---

- special escape sequence character sets:
  - **\d** matches any digit (same as **[0-9]**); **\D** any non-digit (**[^0-9]**)
  - **\w** matches any word character (same as **[a-zA-Z\_0-9]**); **\W** any non-word char
  - **\s** matches any whitespace character ( , \t, \n, etc.); **\S** any non-whitespace
- What regular expression matches dollar amounts of at least \$100.00 ?
  - "**^\\$\d{3,}.\d{2}\$**"

# Thank you!

