中山大學
SUN YAT-SEN UNIVERSITY

Lecture 09. Web Servers & Server Side JavaScript

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

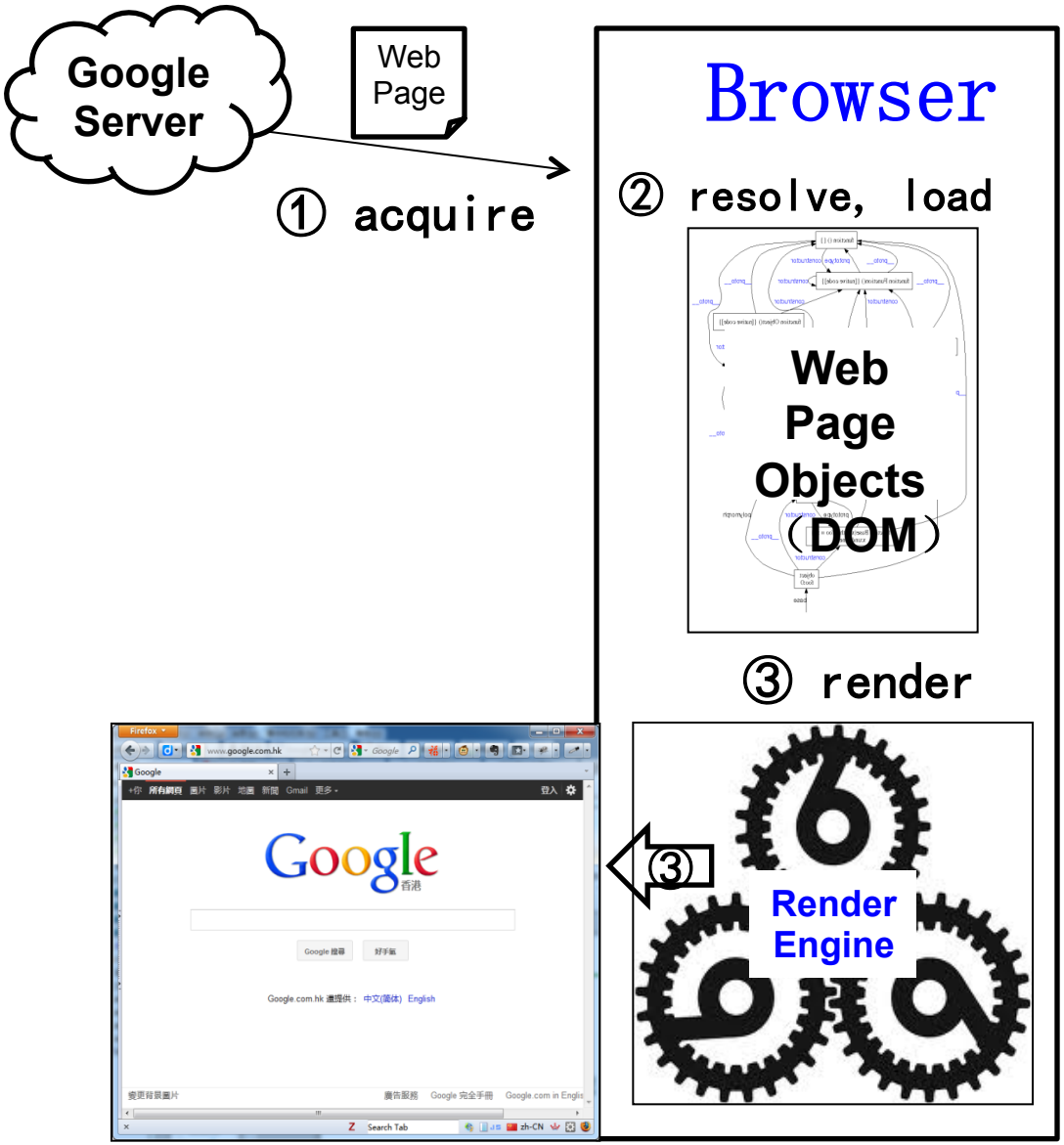
School of Data and Computer Science, Sun Yat-sen University

Outline

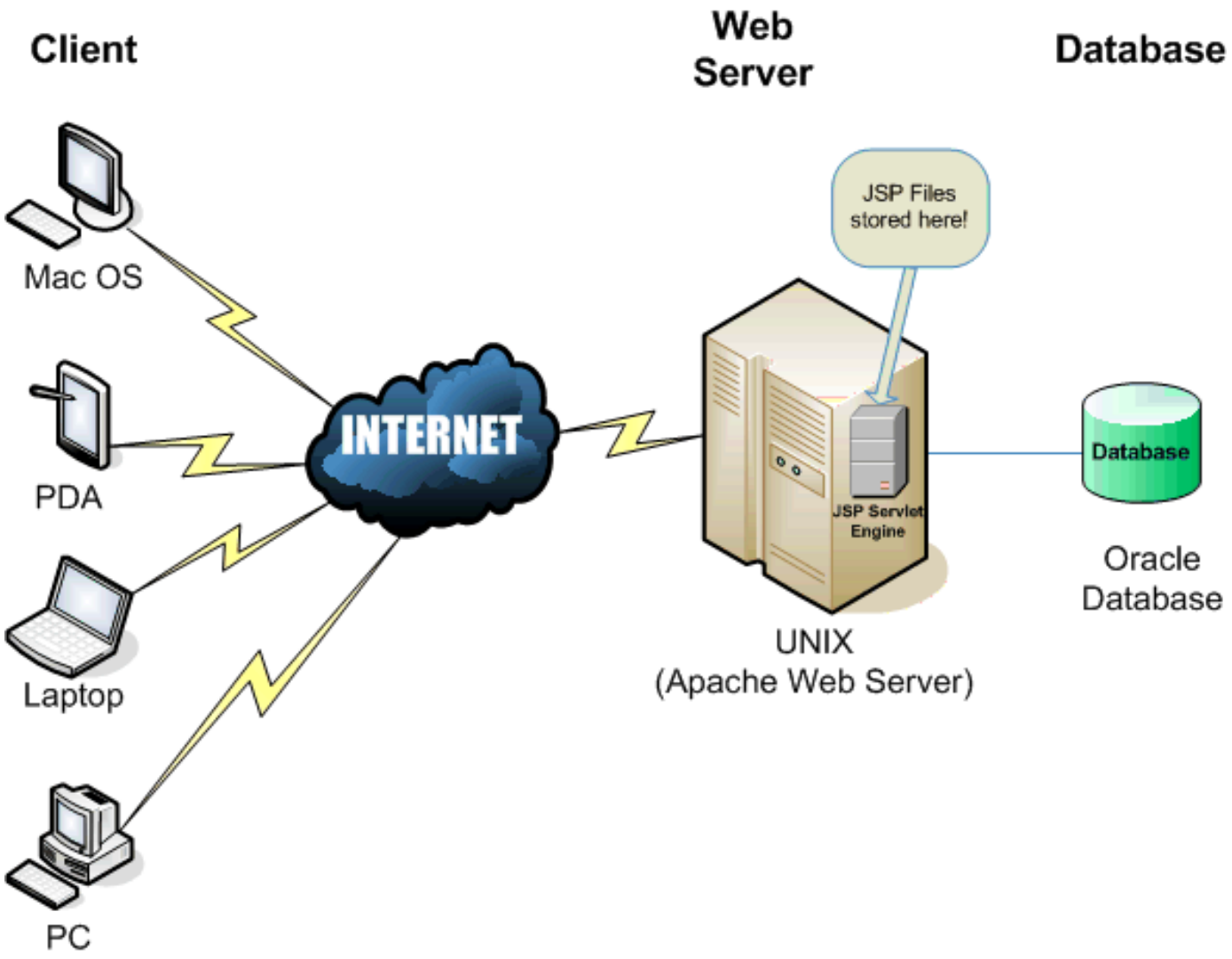
- **Web Servers**
- Node.js in a glance
- Form

Web is a Distributing / Sharing System

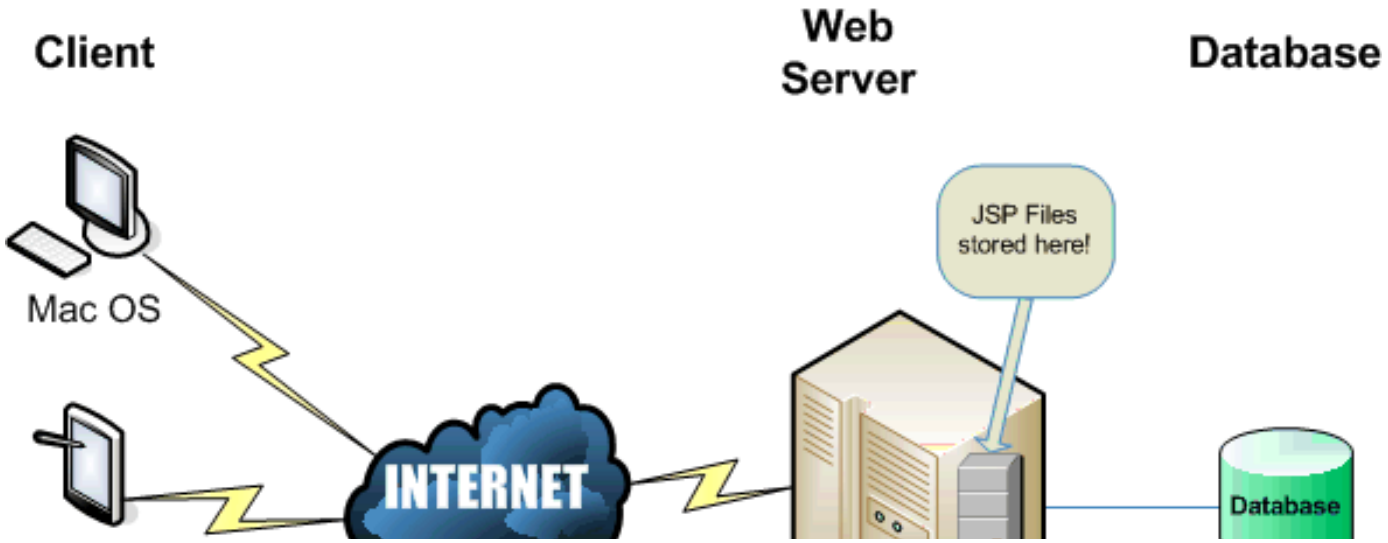
Working mechanism of browsers



Web is a Distributing / Sharing System



Web is a Distributing / Sharing System

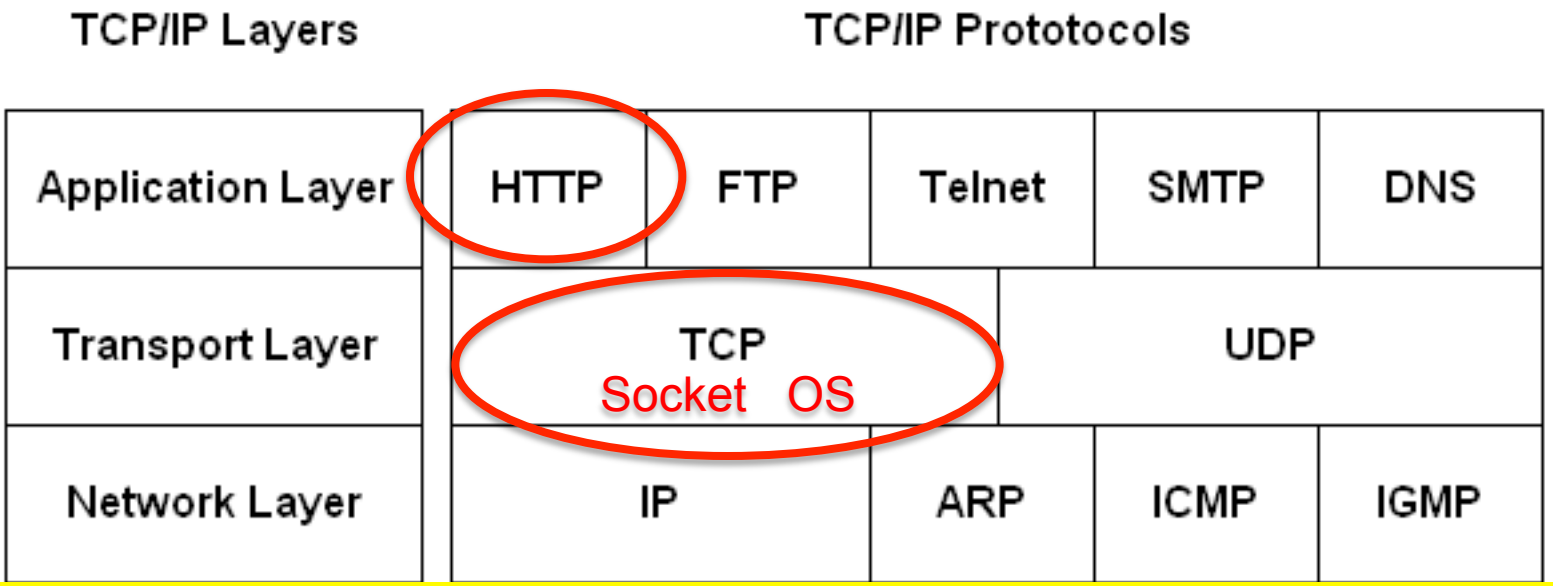


Web servers and browsers

- **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

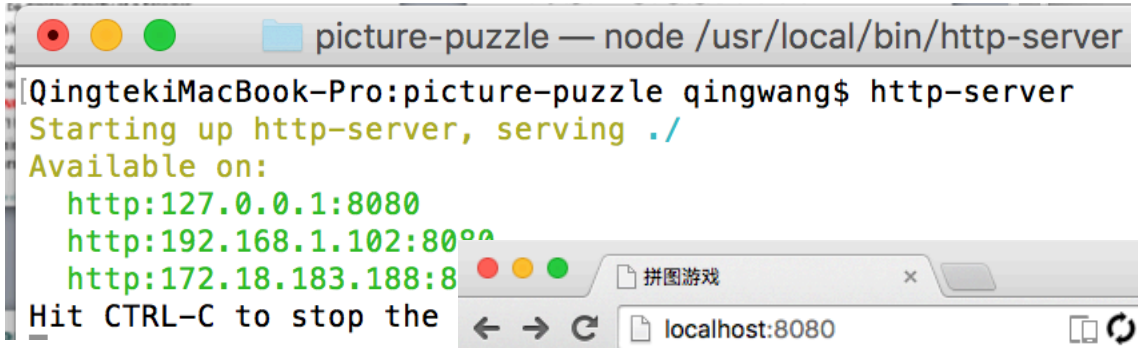
Web Servers

VARNISH CACHE

IIS

G-WAN web server

Share Fifteen Puzzle



拼图游戏

```
picture-puzzle — node /usr/local/bin/http-server — 80×24
QingtekiMacBook-Pro:picture-puzzle qingwang$ http-server
Starting up http-server, serving ./
Available on:
  http:127.0.0.1:8080
  http:192.168.1.102:8080
  http:172.18.183.188:8080
Hit CTRL-C to stop the server
[Mon, 30 Nov 2015 03:17:20 GMT] "GET /" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.3
6"
[Mon, 30 Nov 2015 03:17:20 GMT] "GET /index.js" "Mozilla/5.0 (Macintosh; Intel M
ac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safa
ri/537.36"
[Mon, 30 Nov 2015 03:17:20 GMT] "GET /index.css" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Saf
ari/537.36"
```

拼图游戏

localhost:8080

拼图游戏

Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console | AngularJS

Preserve log | Disable cache | WiFi (30 MB/s 2ms R

Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Other

100 ms 200 ms 300 ms 400 ms 500 ms 600 ms 700 ms 800 ms 900 ms 1000 ms 1100 ms 1200 ms 1300 ms 1400 ms 1500 ms 1600 ms 1700 ms

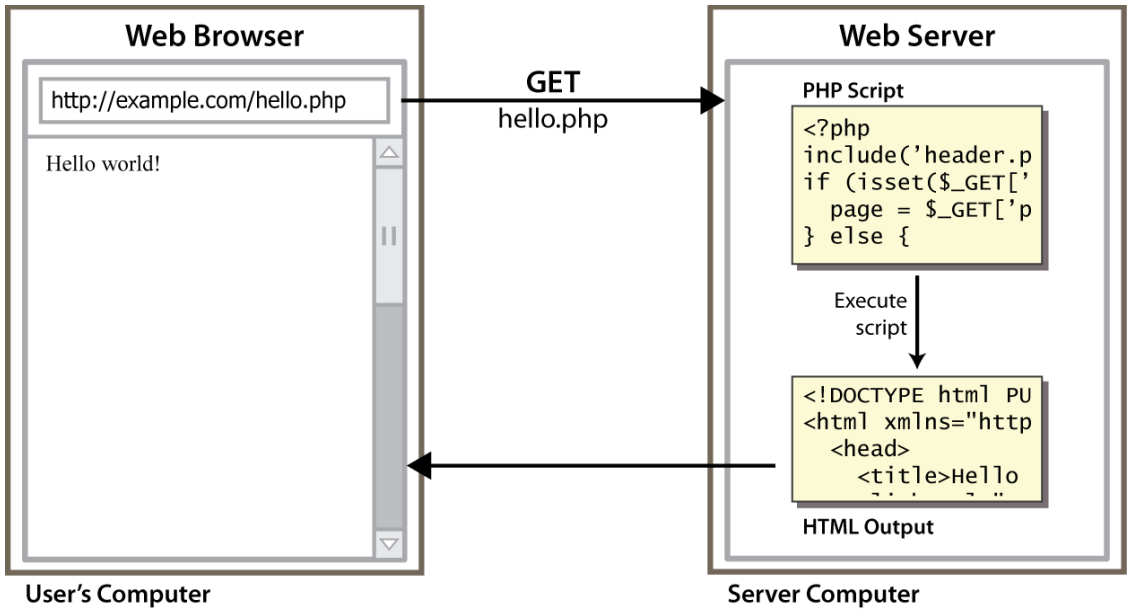
Name

- localhost
- topFrame.js
- Clipper.js
- ContentPreview.js
- Coordinator.js
- GlobalUtils.js
- CustomTooltipEli...
- jquery-2.1.4.min.js
- index.js
- index.css
- checkSimSearch.js
- pageVisible.js
- PageInfo.js
- isTest.js
- domReady.js
- panda.jpg
- FN-close.png
- tab-logo.png

Headers | Preview | Response | Timing

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>拼图游戏</title>
  <script type="text/javascript" src='http://code.jquery.com/jquery-2.1.4.min.js'></script>
  <script type="text/javascript" src='index.js'></script>
  <link rel="stylesheet" href="index.css">
</head>
<body>
  <div id='container'>
    <h1>拼图游戏</h1>
    <div id='puzzle-pane'>
      <div id='1'></div>
      <div id='2'></div>
      <div id='3'></div>
      <div id='4'></div>
      <div id='5'></div>
      <div id='6'></div>
      <div id='7'></div>
      <div id='8'></div>
      <div id='9'></div>
      <div id='10'></div>
      <div id='11'></div>
      <div id='12'></div>
      <div id='13'></div>
      <div id='14'></div>
      <div id='15'></div>
    </div>
    <button>重新开始</button>
  </div>
</body>
</html>
```

Lifecyle 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

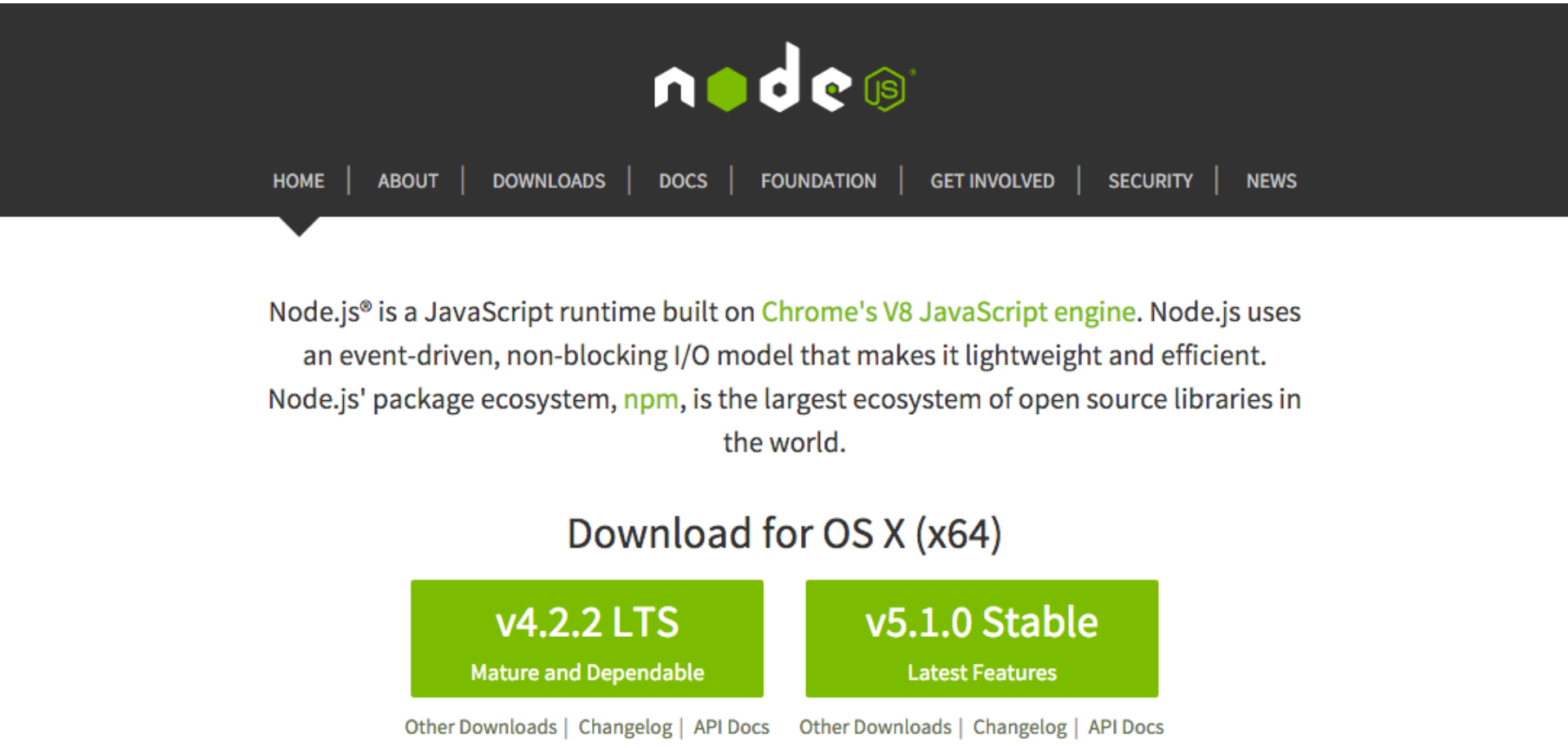
- Static Page
  - Client/Consumer's Viewpoint: an url refers to an identical html file
  - Server/Producer's Viewpoint: a file stored within or sub-within the root folder of a Web Server
  - it is a html …
  - Can be display directly at a browser
- Dynamic Page
  - Client/Consumer's Viewpoint: an url 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

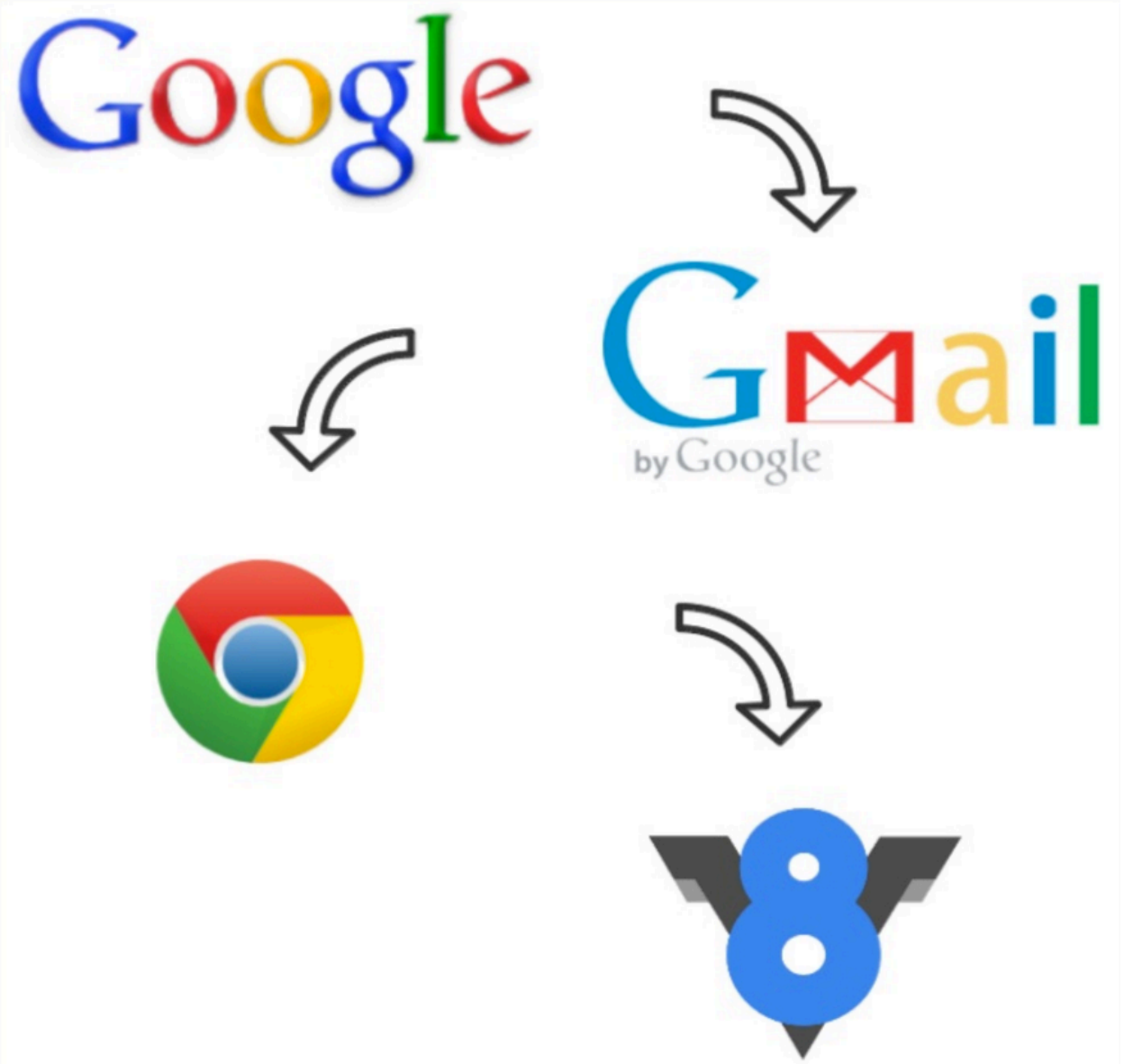
- server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl
- the web server contains module that executes those programs and send back their output as responses to web requests
- each language/framework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

Outline

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

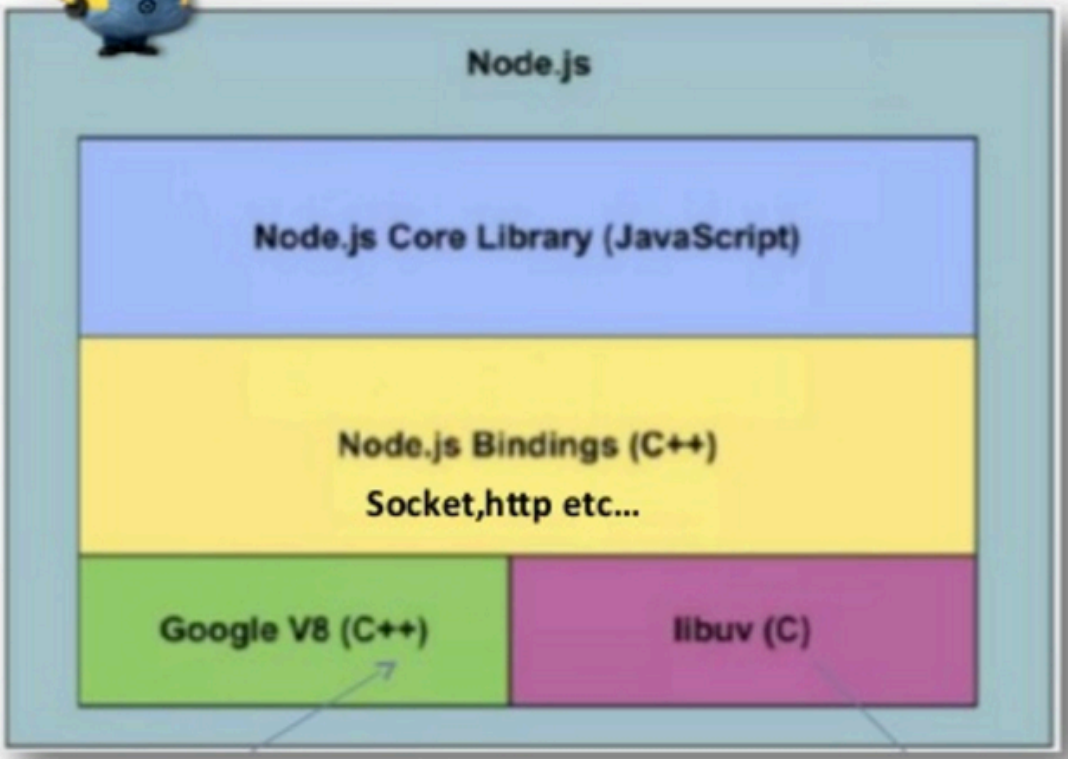


https://nodejs.org


Google
Gmail
by Google

NodeJS under the hood...

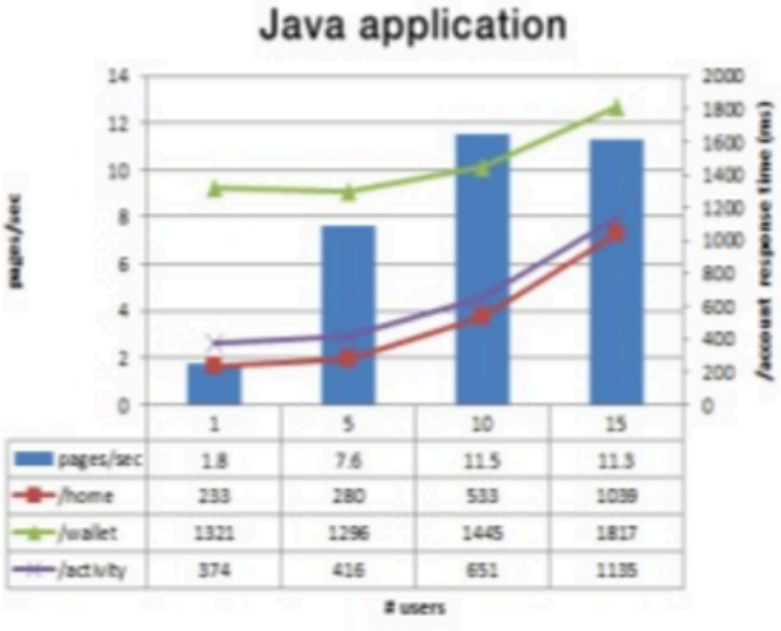
Callbacks + Polling
Epoll, POSIX AIO, Kqueue ...



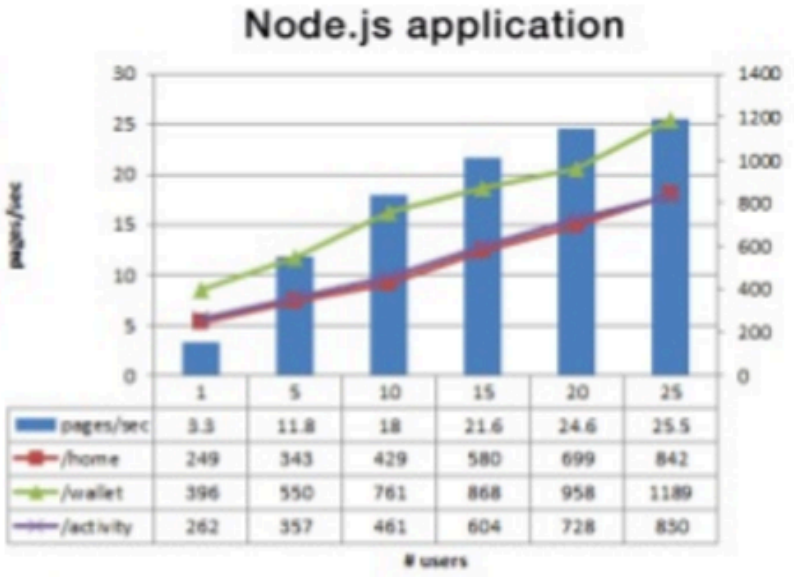
Open source JavaScript engine, platform
Optimizer, JIT, inline caching, GC

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

The PAYPAL STORY



	1	5	10	15
pages/sec	1.8	7.6	11.5	11.3
/home	233	280	533	1039
/wallet	1321	1296	1445	1817
/activity	374	416	651	1135



	1	5	10	15	20	25
pages/sec	3.3	11.8	18	21.6	24.6	25.5
/home	249	343	429	580	699	842
/wallet	396	550	761	868	958	1189
/activity	262	357	461	604	728	850

- **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)

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

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

express
Fast, unopinionated, minimalist web fra...

4.13.1 published 5 months ago by dougwilson

pm2
Production process manager for Node.JS...

0.14.3 published 5 months ago by jshkurti

grunt-cli
The grunt command line interface.

0.1.13 published 2 years ago by tkellen

npm
a package manager for JavaScript

2.13.0 published 5 months ago by zkat

karma
Spectacular Test Runner for JavaScript.

0.13.1 published 4 months ago by dignifiedq...

bower
The browser package manager

1.4.1 published 8 months ago by sheerun

cordova
Cordova command line interface tool

5.1.1 published 6 months ago by stevegill

coffee-script
Unfancy JavaScript

1.9.3 published 6 months ago by jashkenas

gulp

forever

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%!

https://docs.npmjs.com/

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

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.goog
    Google:
    <input name="q" />
    <input type="submit" value
</form>
其它内容.....
```

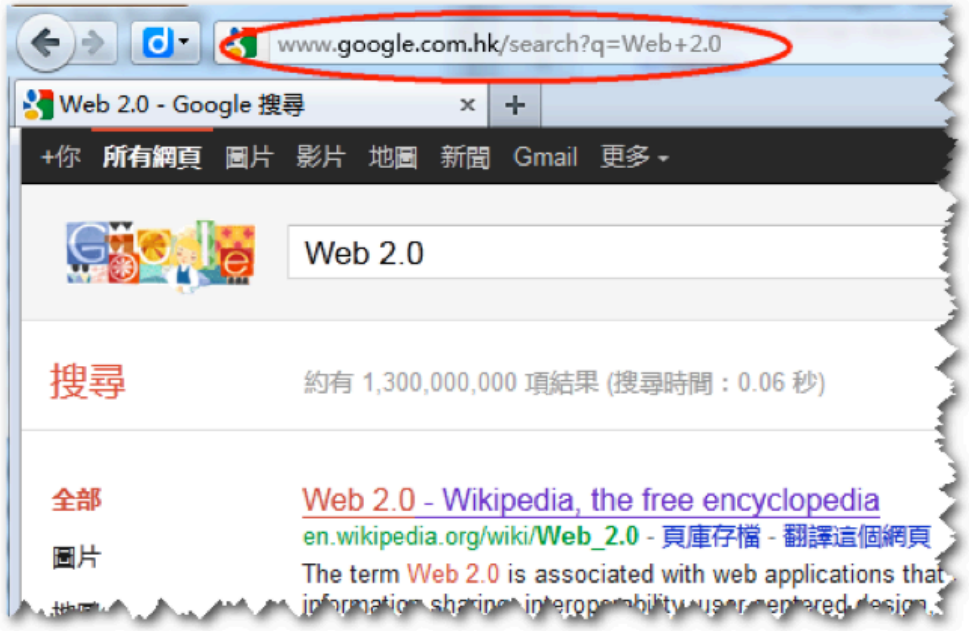
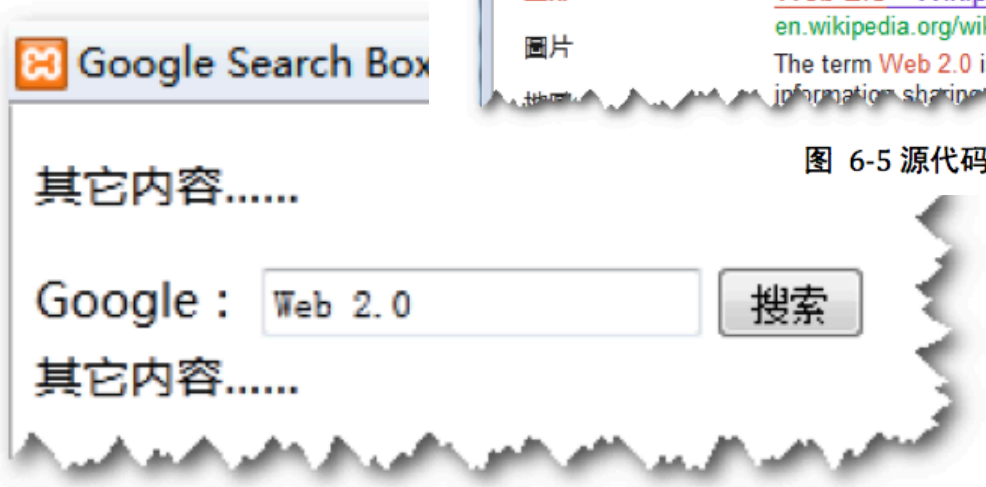


图 6-5 源代码 6-3 运行效果



Form controls

表 6-2 表单控件元素（input）

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

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……
```

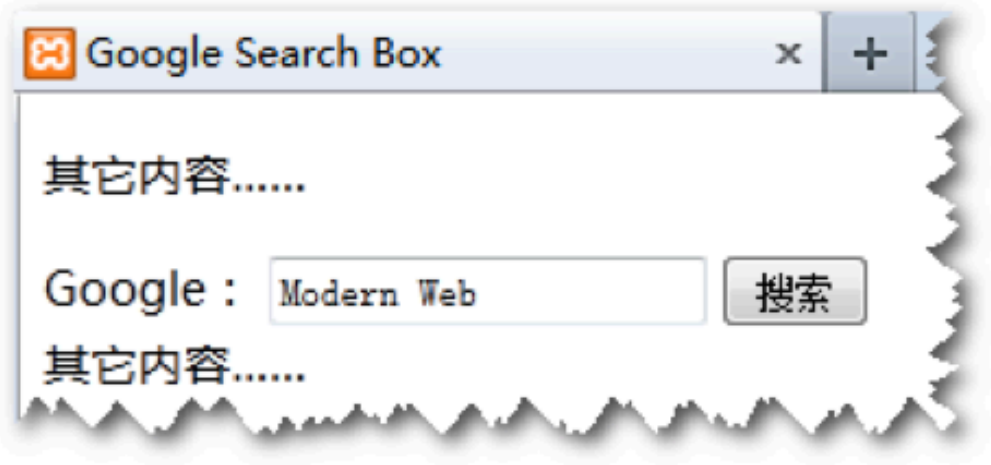
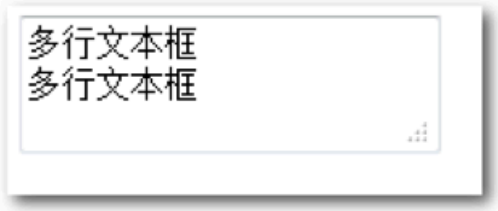


图 6-6 源代码 6-4 运行效果

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） type="text"	
文本域（多行） textarea	
提交、清除按钮 type="submit" type="reset"	
多选框 type="checkbox"	
单选框 type="radio"	
下拉菜单和列表 select	

所有表单控件都是行内元素

必须嵌套在块元素

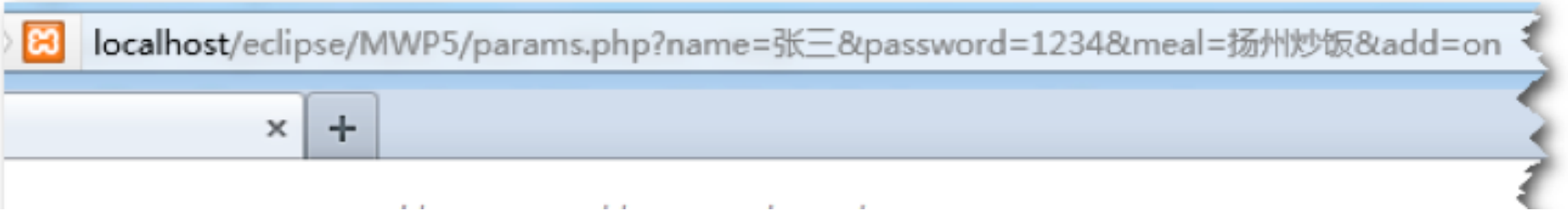
通常是表单（form）内使用

$(...).val 操作表单控件的值

GET vs. POST

图 6-18 订餐表单

method="GET"



GET vs. POST

源代码 6-19 POST 提交表单示例

```
<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
    <input type
</form>
```

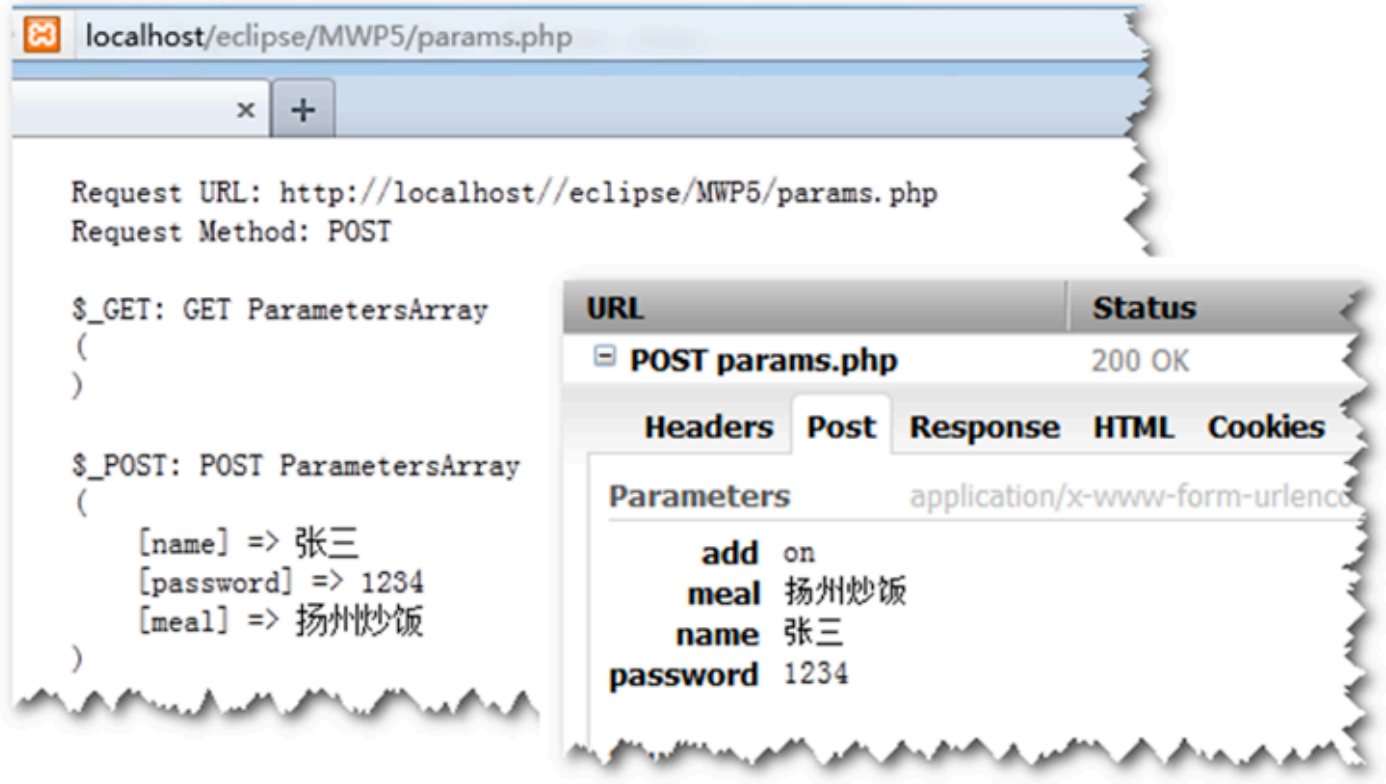


图 6-20 POST 方法提交后的结果网页和 Firebug 显示的 HTTP 报文

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[4]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[5]。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 P
用户在线购买
用户使用刷新
出警示对话框

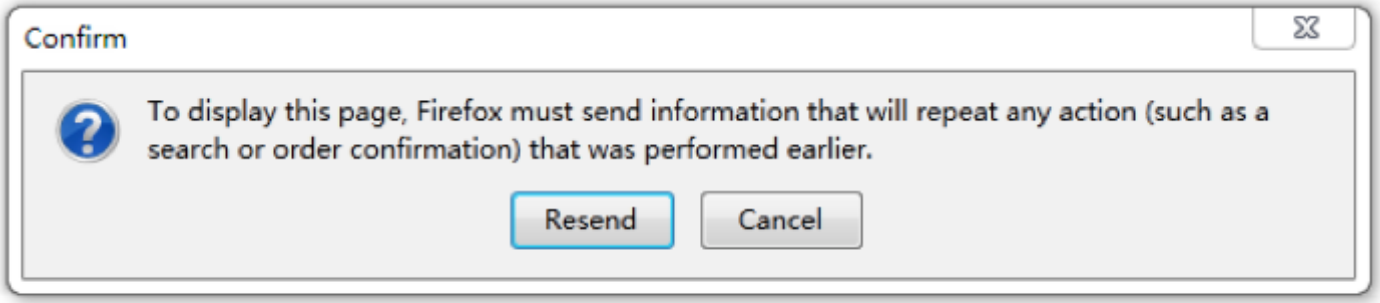
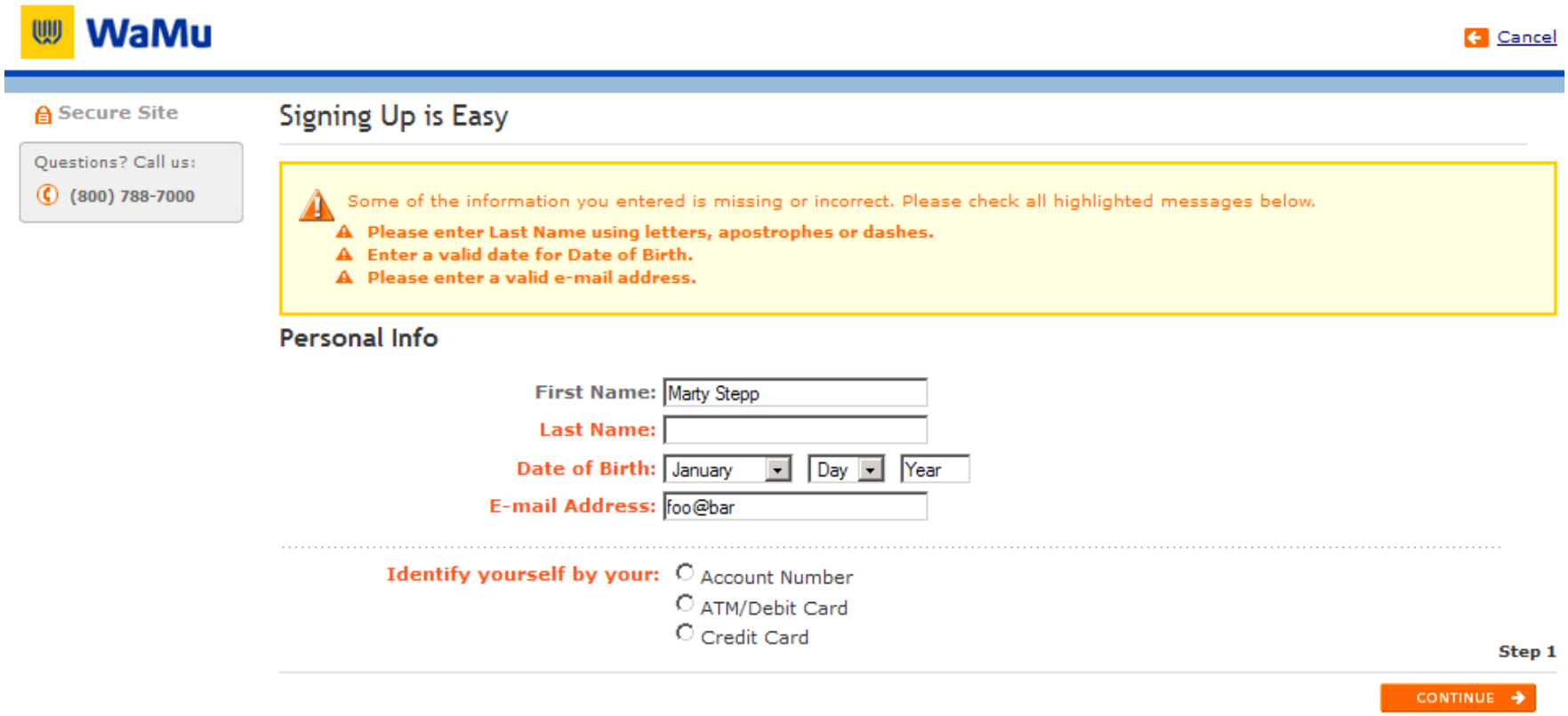


图 6-21 重复提交警示对话框

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



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

Regular Expression

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

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 \/>/" matches lines containing
 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", "Gooogle", "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)?

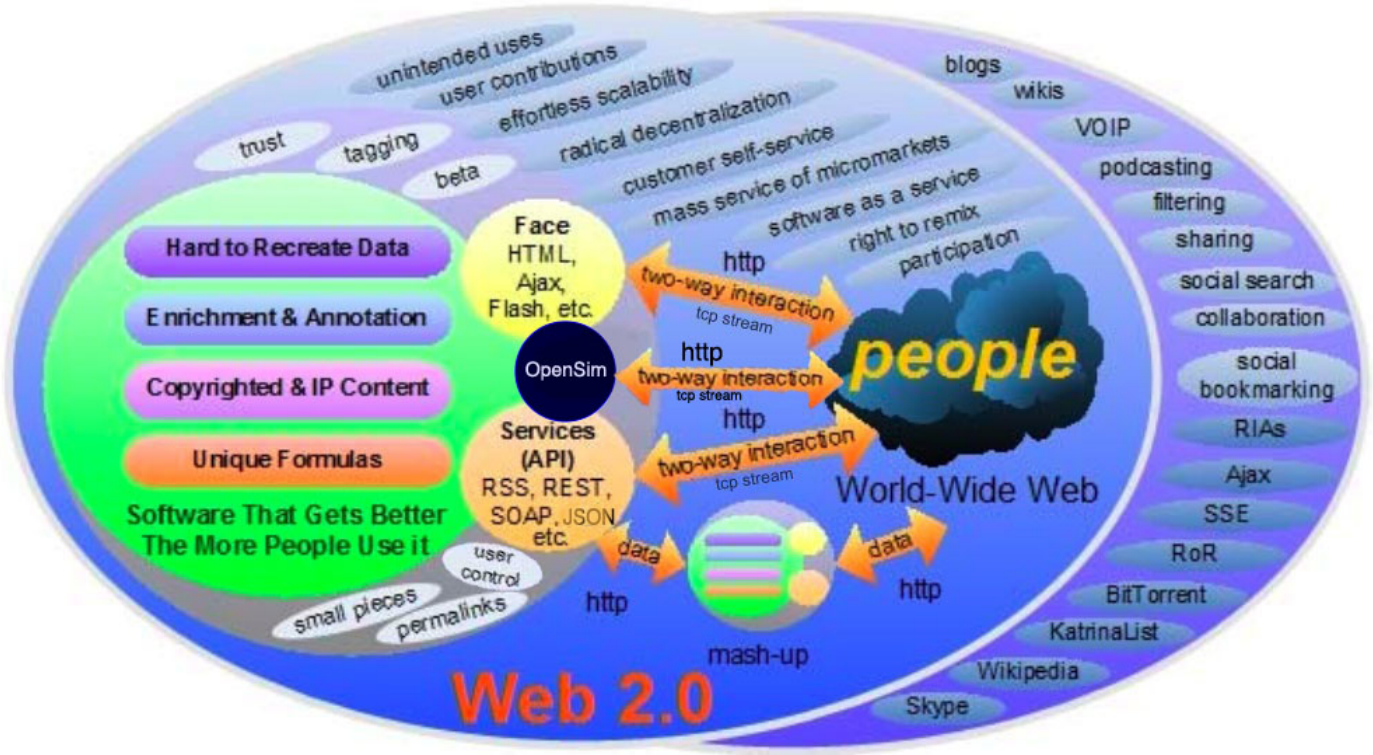
Character ranges: [start-end]

- 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!


unintended uses
user contributions
effortless scalability
trust
tagging
beta
radical decentralization
customer self-service
mass service of micromarkets
software as a service
right to remix
participation
blogs
wikis
VOIP
podcasting
filtering
sharing
social search
collaboration
social bookmarking
RIAs
Ajax
SSE
RoR
BitTorrent
KatrinaList
Wikipedia
Skype
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
user control
small pieces
permalinks
mash-up
Web 2.0