CS 696 Mobile Phone Application Development
Fall Semester, 2009
Doc 14 Network Data
Oct 15, 2009

Ways to transport data on the Network

HTTP

- Web scraping
- Markup languages

SOAP
XML-RPC
RMI

Client-Server Programming

HTTP

Apache HttpComponents HttpClient

http://hc.apache.org/

Java classes that implements Http 1.1 protocol

- Access web pages
- Http headers
- Client-side authentication
- Http state managements
- Connection management

Does not

- Render web page
- Run Javascript
- Handle CSS

Sample Program

```
public class ClientExamples {
    public static void main(String[] args) {
      String url = "http://www.eli.sdsu.edu/courses/fall09/cs696/index.html";
      HttpClient httpclient = new DefaultHttpClient();
      HttpGet getMethod = new HttpGet(url);
      try {
         ResponseHandler<String> responseHandler = new BasicResponseHandler();
         String responseBody = httpclient
                .execute(getMethod, responseHandler);
         System.out.println(responseBody);
      } catch (Throwable t) {
         System.out.println(t);
      }
      httpclient.getConnectionManager().shutdown();
    }
}
```

Output

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
          <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
          <title> CS 696: Course Web Site </title>
</head>
<body bgcolor="FFFFFF">
<table border="0" width="100%">
          <tr>
                    <td width="30">
                              <a href="http://www.sdsu.edu/"> <img border="0" src="http://www.sdsu.edu/graphics/sdsu_logo_3.old"
alt="SDSU" /> </a>
                    </td>
                    <th>
                              CS 696 Mobile Phone Application Development
                              <br />
                              Fall Semester, 2009
                              <br />
                              Course Web Site
          </th>
          <td width="30">
                    <a href="http://cs.sdsu.edu/"> <font color="#FF0000"> DCS</font> </a>
          </td>
</tr>
<tr>
          <td>
          </td>
          <td>
                    <center>
                              <a href="../../index.html"> To Roger Whitney's Course Information </a>
                              <br />
                              San Diego State University -- This page last updated 1-Sep-09
                    </center>
          </td>
```

HttpClient & Android

HttpClient is included in Android

So can access web pages in Android code

But why not just use WebView?

Data

Web scraping

Extracting information from websites

Web scraping

Html sucks as a way to transport data

Don't use it to provide applications with data

Markup Languages for Data

XML
JSON
YAML

XML

XML

XML creators wanted
- Flexibility of SGML
- Simplicity of HTML

1998 - Version 1.0 of XML

Key differences from HTML
- Presentation is separate from document description
- Error Checking
- Unambiguous Structure

Uses
- Share Structured Data
- Encode documents
- Serialize data

XML is about

Document structure

Describing data

```
<?xml version="1.0" ?>
<CATALOG>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>CBS Records</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1988</YEAR>
    </CD>
</CATALOG>
```

Parts of The XML Universe

Basic Syntax

XML 1.0 spec, XML 1.1

XLinks

Namespaces

Markup Languages

XHTML

MathML

SMIL

VoiceXML

Data Addressing & Query

XPath

XPointer

XML Query Language (XQL)

Document Modeling

Document Type Definitions (DTDs)

XML Scheme

Presentation and Transformation

XML Stylesheet Language (XSL)

XSL Transformation Language (XSLT)

Cascading Style Sheets (CSS)

Extensible Stylesheet Language for Formatting Objects (XSL-FO)

Levels of XML

Well-formed

XML document that satisfies basic XML structure

Valid

XML document that is well-formed and

Specifies which tags are legal

A Document Type Definition (DTD) is use to specify

- Legal tags
- Correct tag nesting

Well-Formed XML Documents

Optional Prolog
Root Element

```
<?xml version="1.0" ?>
<!-- A comment -->
<greetings>
      Hello World!
</greetings>
```

```
<?xml  version="1.0"  encoding='iso-8859-1' standalone=no ?>
<!-- A comment -->
<facts>
      <statement>We know 5 &lt; 10</statement>
      <statement><![CDATA[We know 5 <10]]></statement>
</facts>
```

Valid XML Documents

XML document that is well-formed and
Specifies which tags are legal

A Document Type Definition (DTD) is use to specify

- Legal tags
- Correct tag nesting

```
<?xml version="1.0" ?>
<!DOCTYPE greetings [
        <!ELEMENT       greetings    ( from, to, message, date?)>
        <!ELEMENT       from         ( name )>
        <!ELEMENT       to           ( name )>
        <!ELEMENT       message      ( #PCDATA )>
        <!ELEMENT       date         ( #PCDATA )>
        <!ELEMENT       name         ( #PCDATA )>
]>
<greetings>
        <from><name>Roger</name></from>
        <to><name>World</name></to>
        <message>Hi</message>
</greetings>
```

XML Namespaces

An XML namespace is a group of Elements & Attributes

XML namespaces allows mixing of elements from different DTDs

```
<?xml version="1.0" ?>
<whitney:greetings  xmlns:whitney=”http://www.eli.sdsu”
                    xmlns:godot=”http://www.waiting.com”>
    <whitney:from>
            <godot:firstname>Roger</godot:firstName>
    </whitney:from>
    <whitney:to>
            <godot:firstname>John</godot:firstName>
    </whitney:to>
    <whitney:message>Hi</whitney:message>
</whitney:greetings>
```

XML Schema (XSD)

A way to define XML documents

Spec published in 2001

Allow element content to have a type

Allow element content to be restricted

Sample Schema parts

```
<datatype name="Price">
    <scalar datatype="float" decimals="2"/>
</datatype>

<datatype name="MovieTicketPrice">
    <scalar datatype="Price" digits="1"
        maxvalue="9.50" minvalue="1.50"/>
</datatype>

<elementtype name="MovieTicket">
    <model>
        <element name="MovieTitle" type="string"/>
        <element name="TicketPrice" type="MovieTicketPrice">
    </model>
</elementtype>
```

XML Using Schema

```
<MovieTicket>
    <MovieTitle>Gone With the Wind</MovieTitle>
    <TicketPrice>6.50</TicketPrice>
</MovieTicket>
```

XML & Android

Android three XML parsers

W3C DOM
SAX
XML Pull Parser

Android SOAP & XML-RPC

Android does not come with SOAP or XML-RPC

But you can add standard Java implementations if you like

Some Performance

Time in seconds

	Connect time	Send String 21,000 Chars	Send 5,000 integers	Server LOC	Message size sending 100 integers
socket	0.002242	0.001377	6.71	25	85,863
Corba	0.000734	0.004601	1.52	18	27,181
XML-RPC	0.007040	0.082755	100.34	17	324,989
SOAP	0.000610	0.294198	1,324.30	10	380,288

Factor slower/larger than using Socket

	Connect time	Send String 21,000 Chars	Send 5,000 integers	Server LOC	Message size sending 100 integers
Corba	0.3	3.3	0.2	0.7	0.3
XML-RPC	3.1	60.1	15.0	0.7	3.8
SOAP	0.3	213.7	197.4	0.4	4.4

Code written in Python

http://www-128.ibm.com/developerworks/webservices/library/ws-pyth9/

JSON

JSON

http://www.json.org/

JavaScript Object Notation

data-interchange format

rfc 4627

Maps to/from strings

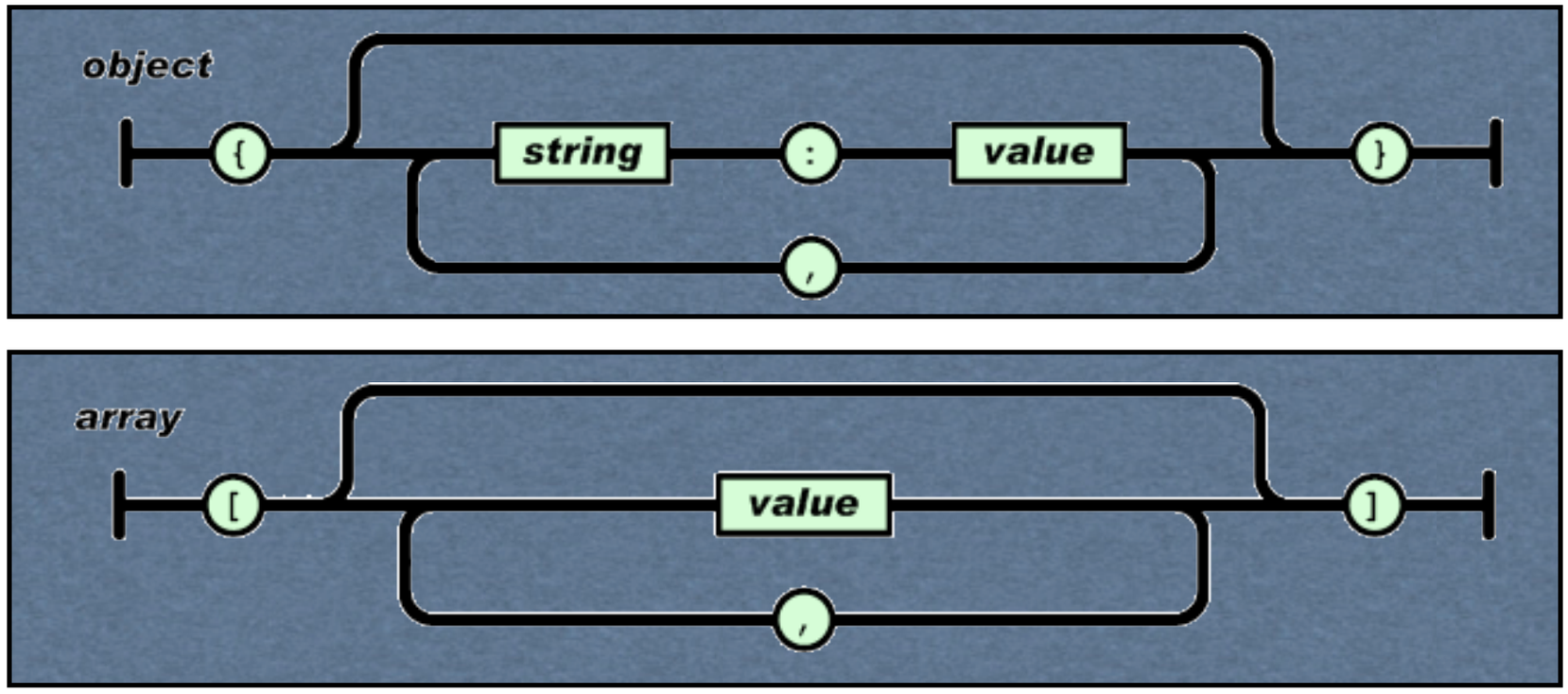
- null
- true, false
- number
- string
- array
- objects

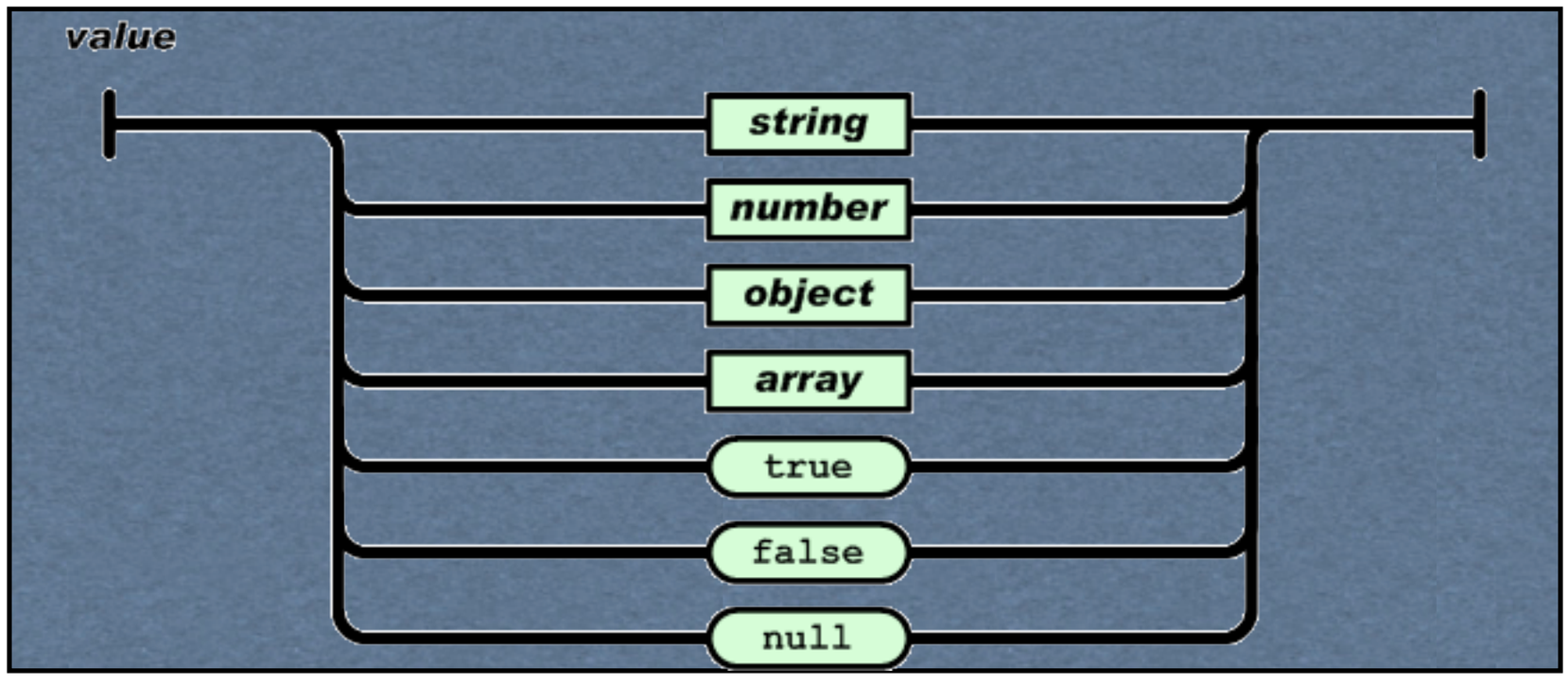
Implementations in

C, C++, C#, D, E, Java, Objective C
Cold Fusion, Delphi, Erlang, Haskell
JavaScript, Lisp, LotusScript, Perl,
PHP, Pike, Prolog, Python, Ruby,
Scala, Smalltalk, Android

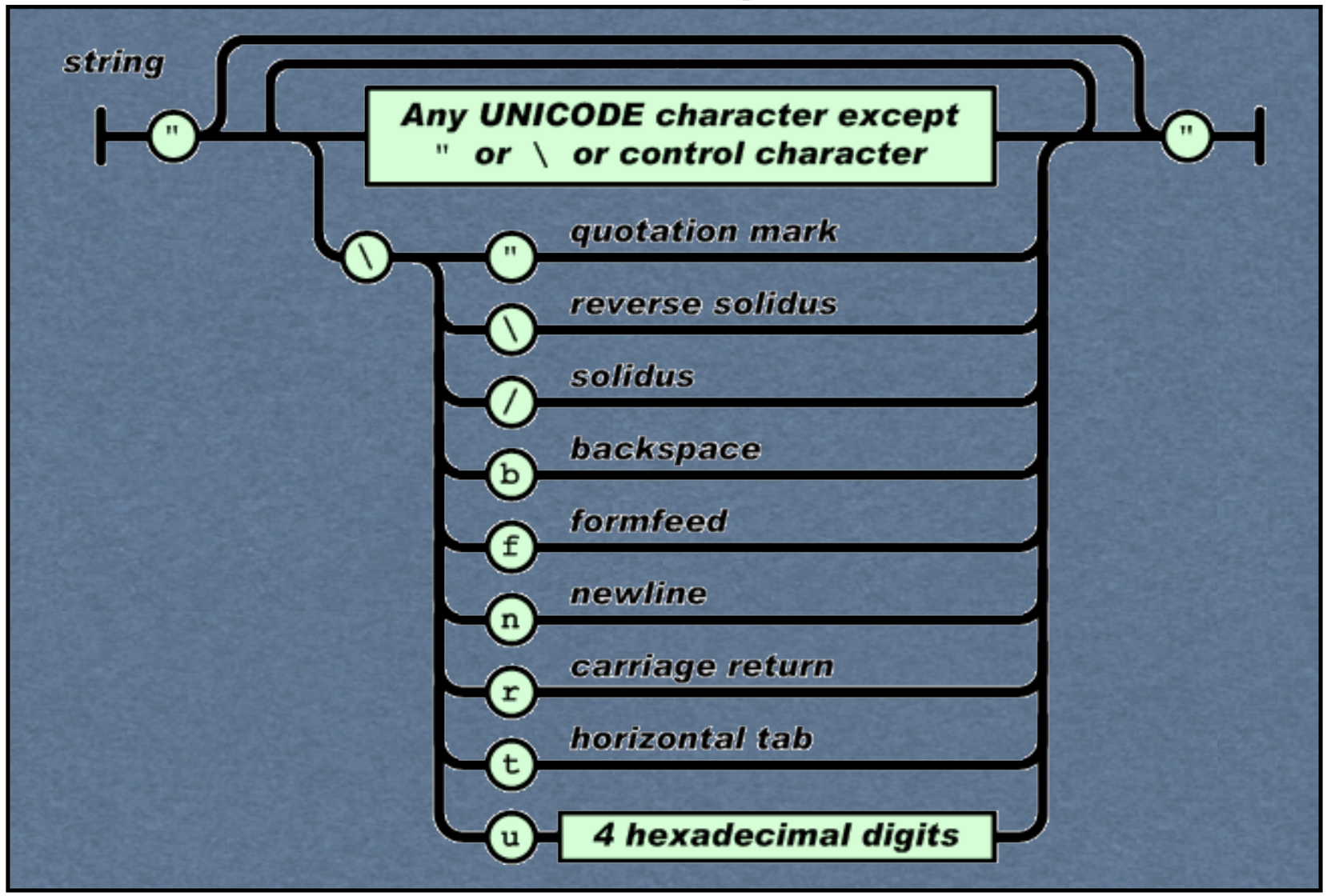
JSON Definition

Source: http://www.json.org/

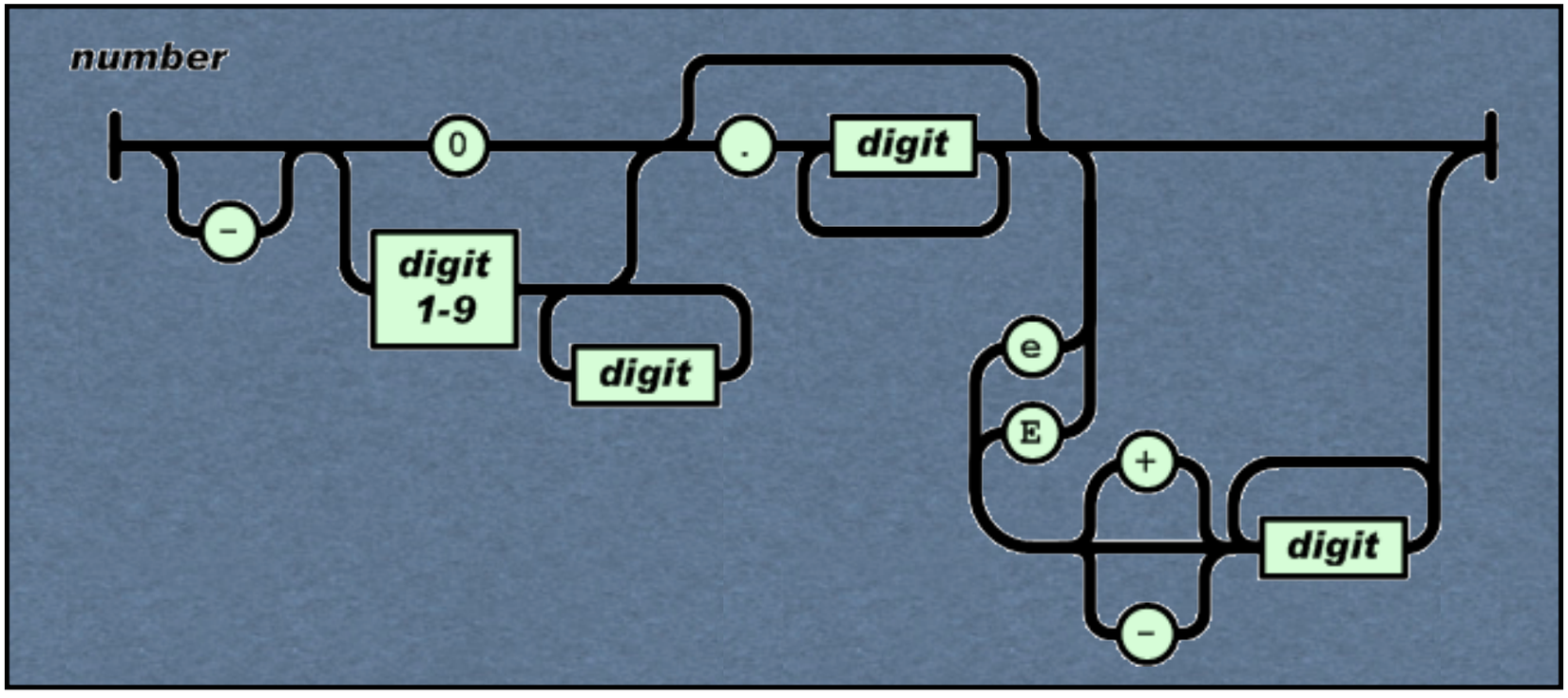




String


string
"
Any UNICODE character except
" or \ or control character
"
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits

Number



Examples

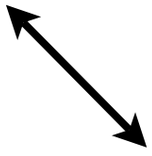
Java Structure

```
Vector array = new Vector();
array.append(new Integer(12));
array.append("Egypt");
array.append(new Boolean(false));
array.append(new Integer(-31));
```

JSON Representation

```
[12,"Egypt",false,-31]
```

```
HashMap<String,Integer> object = new HashMap<String,Integer>();
object.put("lowerBound", 18);
object.put("upperBound", 139);
```



```
{"lowerBound":18,"upperBound":139}
```

Possible Client-Server Usage

Use JSON as protocol syntax

Use JSON libraries to

Generate client-server messages

Parse messages from network

JSON in Java

http://www.json.org/java/index.html

A Java JSON library

JSONObject

- Constructs JSON strings
- Parses JSON strings

```
JSONObject json = new JSONObject();
json.put("lowerBound", 18);
json.put("upperBound", 139);

String objectString = json.toString();       //"{"lowerBound":18,"upperBound":139}"

JSONObject newJson = new JSONObject(objectString);
int bound = newJson.getInt("lowerBound");       // 18
```

JSONStringer

Convience class for creating JSON strings

```
myString = new JSONStringer()
    .object()
        .key("JSON").value("Hello, World!")
    .endObject()
    .toString();
```

```
{"JSON":"Hello, World!"}
```

```
JSONStringer test = new JSONStringer();
    test.array().value(1).value("b").value("3").endArray();
```

```
[1,"b","3"]
```

JSONStringer

```
JSONStringer test = new JSONStringer();
test.array().object();
test.key("firstname").value("Roger");
test.key("lastname").value("Whitney").endObject();
test.object().key("firstname").value("Robert").key("lastname").value(
      "Edwards").endObject();
test.endArray();
test.toString();
```

```
[{"firstname":"Roger","lastname":"Whitney"},{"firstname":"Robert","lastname":"Edwards"}]
```

Sample Usage

```
String url = "http://www.eli.sdsu.edu/courses/fall09/cs696/examples/names.json";
HttpClient httpclient = new DefaultHttpClient();
HttpGet getMethod = new HttpGet(url);
try {
      ResponseHandler<String> responseHandler = new BasicResponseHandler();
      String responseBody = httpclient
   .execute(getMethod, responseHandler);

      JSONArray jsonData = new JSONArray(responseBody);
            for (int k = 0; k < jsonData.length(); k++) {
                  JSONObject person = (JSONObject) jsonData.get(k);
                  String firstName = person.getString("firstname");
                  System.out.println(firstName);
            }
      } catch (Throwable t) {
            System.out.println(t);
      }
httpclient.getConnectionManager().shutdown();
```

"http://www.eli.sdsu.edu/courses/fall09/cs696/examples/names.json"

[{"firstname":"Roger","lastname":"Whitney"},
{"firstname":"Robert","lastname":"Edwards"},
{"firstname":"Kris","lastname":"Stewart"}]

Client-Server

Simple Date Example - Protocol

Client Commands	Server Response
"date" ended by line feed "date\n"	current date ended by line feed "January 30, 2007\n"
"time" ended by line feed "time\n"	Current time ended by line feed "6:58 pm\n"

Server listens for an incoming request

On request

- reads command
- returns response
- closes connection

On client errors - action not specified

Beware

Can only send bytes across network

Client & server maybe different hardware platforms

What is a newline?

End-of-file indicates connection is closed

Server

Basic Algorithm

```
while (true) {
        Wait for an incoming request;
        Perform whatever actions are requested;
      }
```

Basic Server Issues

How to wait for an incoming request?
How to know when there is a request?
What happens when there are multiple requests?
How do clients know how to contact server?
How to parse client request?
How do we know when the server has the entire request?

Java Date Server

```
public class DateServer {
        private static Logger log = Logger.getLogger("dateLogger");

        public static void main (String args[]) throws IOException {
                ProgramProperties flags = new ProgramProperties( args);
                int port = flags.getInt( "port" , 8765);
                new DateServer().run(port);
        }

        public void run(int port) throws IOException {
                ServerSocket input = new ServerSocket( port );
                log.info("Server running on port " + input.getLocalPort());

                while (true) {
                        Socket client = input.accept();
                        log.info("Request from " + client.getInetAddress());
                        processRequest(
                                client.getInputStream(),
                                client.getOutputStream());
                        client.close();
                }
}
```

Java Date Server Continued

```
void processRequest(InputStream in,OutputStream out)
        throws IOException {

        BufferedReader parsedInput =
                new BufferedReader(new InputStreamReader(in));

        boolean autoflushOn = true;
        PrintWriter parsedOutput = new PrintWriter(out,autoflushOn);

        String inputLine = parsedInput.readLine();

        if (inputLine.startsWith("date")) {
                Date now = new Date();
                parsedOutput.println(now.toString());
        }
    }
}
```

This server needs work

Starting the Server

```
rohan 16-> java -jar DateServer.jar
Feb 19, 2004 10:56:59 AM DateServer run
INFO: Server running on port 8765
```

Sample Java Client

```
import java.io.*;
import java.net.Socket;

class DateClient {
        String server;
        int port;

        public DateClient(String serverAddress, int port) {
                server = serverAddress;
                this.port = port;
        }

        public String date() {
                return send("date\n");
        }

        public String time() {
                return send("time\n");
        }
```

Java Client Continued

```
private String send(String text) {
        try {
                Socket connection = new Socket(server, port);
                OutputStream rawOut = connection.getOutputStream();
                PrintStream out = new PrintStream(new BufferedOutputStream(rawOut));
                InputStream rawIn = connection.getInputStream();
                BufferedReader in = new BufferedReader(new
InputStreamReader(rawIn));

                out.print(text);
                out.flush();
                String answer = in.readLine();
                out.close();
                in.close();
                return answer;
            }
            catch (IOException e) {
                return "Error in connecting to server";
            }
        }
    }
```


Bad very bad - using PrintStream for network code.

Running the Client

```
System.out.println("hi");
DateClient client = new DateClient("127.0.0.1", 4444);
System.out.println( client.date());
System.out.println( client.time());
```

Issue - Avoid Small Packets

```
OutputStream rawOut = connection.getOutputStream();
PrintStream out = new PrintStream(new BufferedOutputStream(rawOut));
```


Use buffers

Issue - Actually Send the request

```
out.flush();
```

Issue - Client will not work on all platforms

```
String answer = in.readLine();
```


What ends a line?

Don't Do this

```
String answer = in.readLine();
```


I did it to keep the example small. One can not get much code on a slide using 24 point font. Plus the Ruby example is sorter than this.

Issue - Close the connection when done

```
out.close();
in.close();
```