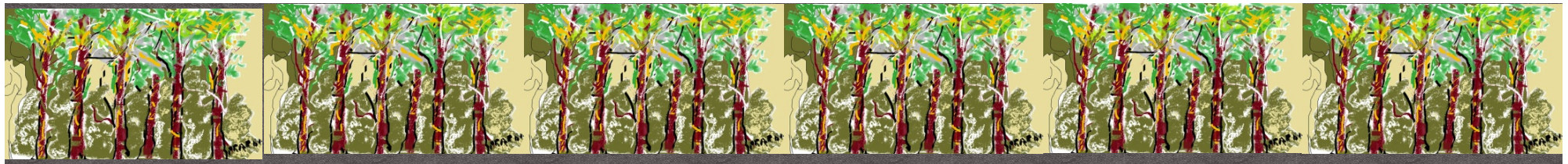


Chapter 12 – Web and Multimedia Programming

Introduction to Programming Languages

First Edition, 2013

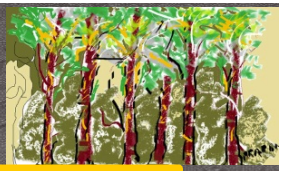
Author: Arvind Bansal

ISBN: 978-146-6565142

Topics Covered

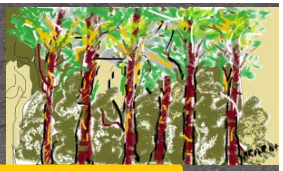
- Code and data mobility issues
- Web based programming
  - XML ; web scripting; applets
  - Security issues
- Virtual machines and runtime interface
  - JVM and Just-in-time compilation
- Components of multimedia systems
  - Representation; transmission; perceptual distortion; synchronization
- Multimedia programming constructs
  - Synchronization constructs
- Abstractions and programming in ALICE, SMIL, Javascript and C#
- Summary

Introduction



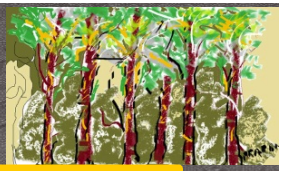
- Since 1990 Web has been established
  - Web is a network of information nodes for resource sharing
  - Resource could be file, database, computing power, or memory
  - In a web resources can be accessed from anywhere
- Capabilities of information nodes
  - Performing computation
  - Storing and retrieving information from an information node
  - Requesting data or code to be transmitted from a remote node
  - Transmitting data or code to a remote node
  - Setting up connection with remote nodes for code and data transfer. The remote node location is called URL (Uniform Resource Locator)
- Availability of computing resources has started the need for code and data mobility
  - Code mobility allows procedures to be executed on remote processors
  - Data mobility allows data to be processed remotely
  - Web programming intertwines code and data mobility for the integration and sharing of computation, multimedia, database and visualization

Applications and Major Issues



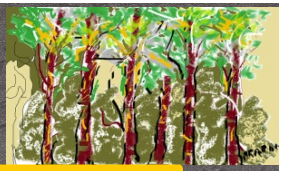
- Almost all major areas such as
  - Banking, telephony, transportation, stock market investments, sale of consumer products, entertainment industry such as game playing and demand based movies, information archives and exchanges education industry, collaborative design and modeling
- Web programming has improved our productivity
  - Reliable communication and sharing of multimedia instead of text
  - Movies, clips, business transactions, E-books, education is available any where and any time enabling globalization
  - Automated indexing software has given us ubiquitous search engine
- Major Issues
  - Distributing the workload to avoid bottlenecks
  - Avoiding data congestion and communication overhead
  - Handling inefficiency of packing and unpacking data
  - Providing security against mal-software, servers, or information leak
  - Handling heterogeneity of architectures and operating systems
  - Developing web based constructs and languages

Code and Data Mobility



- Models of mobile computing
  - *Client-server* ; *remote evaluation*; *code on demand*; *migrating agents*
- Client server model
  - Requesting node is called *client*, and the provider node is called *server*
  - Server performs computation, and sends the result back to the client
  - No mobility of the code
- Remote evaluation
  - Data is transmitted from a web-node N^C to another web-node N^R.
  - N^R has the resources and code to process the data.
- Code on demand; example: Java applet
  - The client node N^C has the resources to execute mobile code and data.
  - The node N^C requests mobile code from the remote node N^R.
  - The mobile code is executed using local resources and data.
- Migrating agents
  - The code may be sent from one node to different web-nodes.
  - The mobile code uses resources of the remote nodes.
  - Mobile code may migrate from one remote node to another.

Handling Heterogeneity

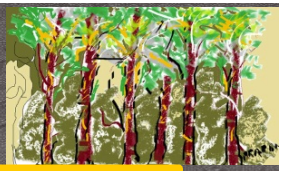


- Heterogeneity problem is caused by
  - Difference in computer architecture
  - Difference in hardware configurations such as available memory and performance capability of the machines
  - Difference in operating systems and their versions
  - Difference in the available compilers to compile the mobile code
  - Difference in the available system and language libraries
- Approaches to solve the problems of heterogeneity
  - Interoperability to translate the format of source to destination node. One machine has to be aware of other machine's format for the translation
  - Virtual machine implementation to enforce homogeneity of environment. Uses a common abstract instruction set
  - Resource availability problem is solved if the source node is aware of remote nodes' capabilities and load conditions
- Compatibility problems between software libraries is a serious problem.

Execution and Migration Efficiency

- Use Just-in-time compiler to transform into efficient native machine code
- Use cache to reduce data transfer overhead;
- Use stream based transmission of multimedia objects so that rendering of video and audiovisuals at the client end can be interleaved with transmission
- Adjusting the resolution at runtime based upon the communication channel capability and traffic congestion
- Keep multiple copies of the frequently used code and data at various nodes called *mirror-sites*.

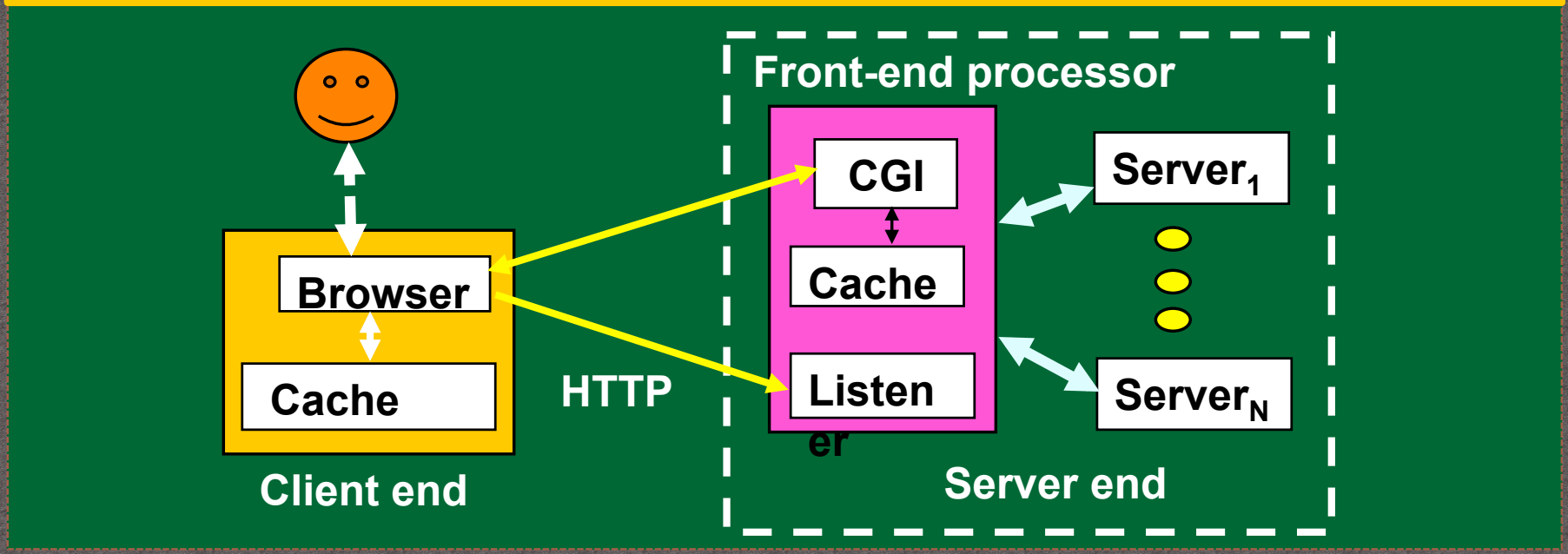
Handling Safety



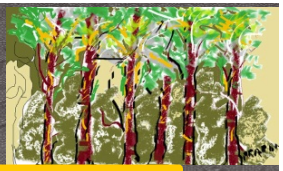
- Issues
  - Mobile code corruption on remote nodes or during transmission
  - Server malfunction due to malicious mobile code
  - Privileged mobile data can be stolen
  - Phishing - a malicious web-site may disguise itself as a genuine web-site
- Safety has handled at multiple levels
  - 1) At the client-end; 2) During transmission; 3) At the server-end; 4) verifying the identity of the client and the mobile code
  - Active mobile code is **sandboxed** – put in a protected area with little access to system area and file structure
  - The transmitted data is encrypted at the source-computer and decrypted at the destination computer
  - Each packet carries the identity of the originating web-node and intermediate web-nodes to ensure that it did not pass through known malicious nodes

Client-server Model of Web Browsing

- The client would send a request by clicking an embedded hyperlink.
- The data would be transmitted from the server to the client.
- The browser at the client-end displays the information to a user.
- The HTML browser has decoders to display various media objects.
- The client carries a cache to archive data of the recently visited sites to avoid excessive data transfer.
- Common Gateway Interface (CGI) reformats the retrieved data.

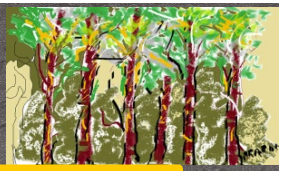


HTML – Browser Language



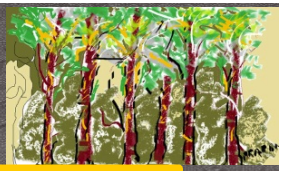
- HTML is the earlier version of popular browser language
  - Uses built-in libraries for human comprehensible visualization
  - Three major components: *head, body,* and *hyperlinks*
  - Head part includes the *title*, *transmission protocol name*, *format information*, *author information*, and *keywords for the search engines*
  - Body part contains annotated document
- Transmission protocol
  - *http* – for web based browsing of data
  - *ftp* – used in file transfer without visualization
  - *file* – denotes file on local computer
  - *telnet* – used for remote login
  - *tel* – used for telephone dialing
  - *modem* – for modem based connection
- Hyperlinks are embedded in inside the documents
  - Locater: *<transfer-protocol>*: // *<node-name>*: [*<port-name>*]/*<path-name>*/*<resource-name>*
  - File resource for media: *<file-name>*.*<media-format-type>*.

Example of HTML



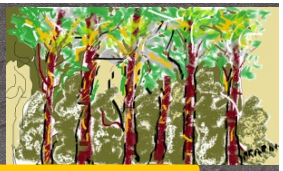
```
<HTML>
<Head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
<META NAME="Author" CONTENT="Arvind Bansal">
<META NAME="Keywords" CONTENT="programming languages, text book”>
 <TITLE> Programming Language E-book</TITLE>
</Head>
<Body BGCOLOR = "#F2FFFF">
<H1 ALIGN = CENTER > Introduction to Programming Languages </H1>
<TABLE border = 0  cellspacing = 0 cellpadding = 2 bottommargin = 0 topmargin = "0"
bgcolor = "white">
<TR>
<TD VALIGN = TOP ALIGN = LEFT> <img SRC = "book.jpg" >
<TD VALIGN = TOP Align = Right> <A HREF = "http://www.crcpress.com"> CRC Press </
A>
</TR>
</Table>
</Body>
</HTML>
```

HTML Limitations and XML



- Limitations of HTML
  - Has fixed tags, and does not support user defined tags.
  - Tags and attribute values are mixed making style change difficult.
  - HTML does not support computation and text processing.
  - HTML has limited capability of any action based upon events.
  - e-commerce organizations need lot of effort to reformat the web-site
- XML
  - Supports user defined tags
  - Supports integration with computation and communication
  - Supports web based scripting and database integration
  - Supports data mobility and code mobility
  - XHTML integrates HTML browser capability and XML's user defined tag
- Major advantages of XML in modeling 3D media objects
  - Graphs based structure can be embedded in XML
  - User defined media object can be represented using XML
  - Can flatten nested structure of database and media objects

XML Representation



- XML description has two parts: DTD and document
- Features of DTD (Document Type Definition)
  - Defines the structure of the documents described in EBNF form
  - DTD description can be associated with an identifier that can be later used
  - DTD can used and loaded as a separate file
- Example of an XML documents

```
<?xml version= 1.0?>
<!- -  comment - DTD description starts here - ->
<! DOCTYPE book  SYSTEM  “book.dtd” >
<!- - comment – document description starts here - ->
<book>
   <author>  John Doe </author>
   <publisher> XYZ-publisher </publisher>
   <year> 1980 </year>
</book>
```

A Simplified Grammar for XML

<*dtd-definition*> **::=** <*element-dtd*> | <*attrList-dtd*>

<*element-dtd*> **::=** '**<!**' **Element** <*element-name*> (<*element-type*> |
EMPTY | **ANY** | '(' **#PCDATA**')') '**>**'

<*attrList-dtd*> **::=** '**<!**' **ATTLIST** <*element-name*> { <*attrName-Type-Default*}+ '**>**'

<*element-type*> **::=** '**(**' <*type*> {(',' | '|') <*type*>]* '**)**' | <*identifier*>

<*element-name*> **::=** <*identifier*>

<*attrName-Type-Default*>**::=** <*attrName*> ' '<*attrType*>' '

<*default*><*attrName*> **::=** <*identifier*>

<*defaultValue*> **::=** '"' <*value*> '"' | **#REQUIRED** | **#IMPLIED** | **#FIXED** <*value*>

<*attrType*> **::= CDATA** | {<*entity*> ' ' }* | <*enumeration*> | <*identifier*> |
{ <*idRef*> ' '}+ | {<*NMToken*> ' '}+ | **NOTATION**

<*entity*> **::=** <*general-entity*> | <*parameter-entity*> | <*embedded-media*>

<*general-entity*> **::=** '<!' **ENTITY** <*entity-name*> <*visibility*> <*definition*> '>'

<*parameter-entity*> **::=** '<!' '%' **ENTITY** <*entity-name*> <*visibility*> <*definition*> '>'

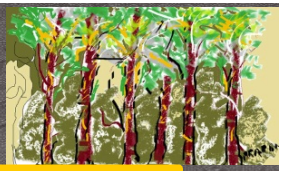
<*embedded-media*> **::=** '**<!**' **ENTITY** <*entity-name*> <*visibility*> <*definition*> **NDATA**
<*format*> '**>**'

<*idRef*> **::=** <*identifier*>

<*format*> **::= .jpg** | **.gif …**

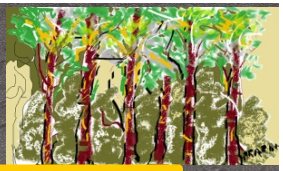
<*visbility*> **::= PUBLIC** <*FPI*> | **SYSTEM**

Style Sheets



- Used to separate the attribute values from the document
  - Changing the style sheets alters the attribute values.
  - Style-sheet carries multiple options for the attribute-value pairs.
  - Needs two files: (1) XML document; 2) a style sheet description.
  - Style sheet description is relatively smaller than the document.
- XSLT
  - A program to transforms an XML program to another using style sheet
  - Transformation is applied on the template using the command '<' **xsl: apply-templates** '>' after the text document.
- Mechanism of changing the style
  - Style sheet contains association of identifiers with attribute values.
  - Document uses the same identifier for the attributes.
  - Different style sheets associate different attribute values with the same identifier.
  - Templates about the style sheets are stored in the head area.

Modeling Complex 2D Objects



- Modeling complex 2D objects
  - Decomposed object into multiple homogenous regions
  - A homogeneous object is modeled as a (polygon, attribute-value pairs)
  - Polygons are represented as a sequence of vertices with an edge
  - Example

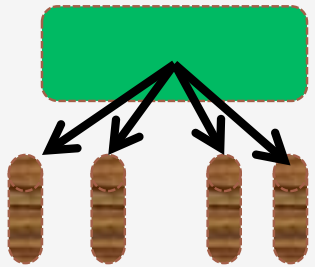
```
<complex-region>
        <region name="Ohio" points=3  color = "red">
                <elements>  x-value  y-value </elements>
                <elements>  x-value  y-value </elements>
                <elements>  x-value  y-value </elements>
        </region>
</complex-region>
```

- Modeling complex 2D image
  - Complex images are modeled as a group of images each with a center of gravity and image file.

Modeling Complex 3D Objects

- Attributes
  - *Shape*, *texture*, *luminosity*, *connectedness to other subpart*
  - Represented as a list of attribute-value pairs in each node
- Representing homogeneous regular shapes
  - Homogeneous objects are sphere, cone, cylinder, prisms
  - Sphere: characterized by center of gravity and radius
  - Cube: coordinates of the leftmost point and length of the side
- Modeling complex 3D objects
  - Model as graph: vertices another 3D object; edges connect the objects
  - **Example**

    Table is a tree with root associated with a rectangular prism and four leaf nodes associated with cylinders.



- VRML (Virtual Reality Markup Language)
  - Uses homogeneous objects to make complex objects.
  - Can stretch or contract regular objects.
  - Can model textures, luminosity, source of light, stereoscopic sound etc.

Embedding Computation in XML

- Two ways to incorporate control abstractions
  - Interface with code written in a web based programming language.
  - Annotate the control abstractions as tags, and features of the control abstraction as attributes of the tags.
- VoiceXML
  - A language for archived telephone conversation
  - Uses tags for variables and control abstractions.
  - **Example**

    '<' var name = <*identifier*> **expression =** <*url*> **'>'**

    '<' **if cond** = <conditional-expression> '<' audio src = <url> '/>'

    <else/> '<' audio src = <url> />

    '<' /**if** '>

Embedding Communication in XML

- Soap (Simple Object Access Protocol) is a protocol to exchange information between two Internet based applications
  - SOAP messages uses XML format without DTD
  - Header tag: <soap:header>; body tag: <soap:Body>
  - Soap message is embedded in <soap:Envelope> and </soap:Envelope>
- Components of <soap:body>
  - URL name where the message is being sent
  - Optional **<soap:Fault>** block to handle the faults
  - **<soap:Fault>** has four elements 1) **<faultcode>** to detect faults; 2) **<faultstring>** - an explanation of the fault; 3) **<faultactor>** - what caused the fault; and 4) **<detail>** - application specific description.

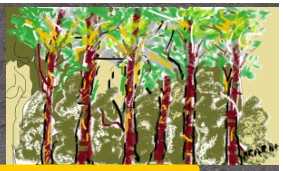
Web Scripting

- Web scripting languages provide computation capability to XML.
- Java provides mobile code capability through *applets.*
- Capabilities of web scripting languages
  - Create a web document on the fly.
  - Modify the web document based upon user interaction or an event.
  - Interact with the user to fill up a form.
  - Provide safety in user interaction.
  - Provide data and control abstraction.
  - Interface with databases such as SQL or Microsoft Access.
  - Change the style of the text.
  - Create and access abstract graphics objects such as radio buttons.
- Some web scripting languages also provide
  - Concurrent constructs to render multiple streams simultaneously
  - Time-delays between streams to provide causality between the events

Integrating XML and Webscripting

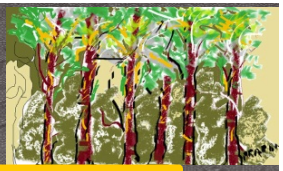
- Approaches
  - Extend XML: XAML (eXTENSIBLE Application Markup Language)
  - Interface with existing languages such as PHP, Javascript or Java
- XAML integrates Microsoft graphics toolkit and XML
  - Special tags to invoke the methods for drawing graphics.
  - **<button** x:name = "btnExit" **Height =** "50" **Width =** "50" **Background =** "white" **Foreground =** "Red" **Content =** "Exit" **click =** "btnExitApp" **/>**
- Interfacing with Javascript
  - A Javascript function can be embedded in the client side XML code.
  - Invoked in response to an event such as mouse-motions
  - Used for: 1) client-end user interactions; 2) simple authentication of the user at the client end; 3) form checking at the client-end; 4) providing simple code such as displaying a clock or a calendar on the webpage; 5) changing the display style of HTML code; and 6) by interfacing with the common gateway interface
- AJAX (Asynchronous Javascript and XML)
  - Integrates Javascript with XML (or XHTML) for dynamic display and user interaction
  - Has two functions: 1) transforming Javascript call to HTTP request; 2) transforming XML data coming from the server-end to XHTML + CSS data

Java Applets I



- Mobile code that migrates to the client-end on demand
  - Compiled on host using JIT compiler or interpreted on JVM
  - Migrates to client end when the webpage containing the applet is loaded
  - Goes through code verification, and is *sandboxed* for security
  - JVM runtime environment is needed to visualize applet in Java
  - Java applets imports class libraries – *java.applet.Applet* and *java.awt.Graphics*
- Major methods
  - *Init method* : executed when an applet is invoked
  - *Start method* : executed after return to the page containing the applet
  - *Stop method* is activated when the user moves to another page
  - *Destroy method* is executed when the browser shuts down
  - *Paint method* is used to repaint media objects during applet's execution
- Attributes
  - Codebase, code, object, archive, alt, name, and media layout primitives such as width, height, align, vspace and hspace
  - Applet uses optional tag **'<' param name=** <*identifier*> **value=** <*value*> **'>'** embedded inside the HTML or XML document to pass the parameter values
  - Parameters declared in XML code are pulled in the applet using a method **getparameter**(<*Parameter-name*>).

Java Applets II



- Abstract syntax

  <*Applet-tag*> ::= '<' **Applet**

  [**codebase =** <*URI*>] **code =** <*Applet-file*> [**alt =** <*Alternate-text*>]

  [**name =** <*AppletInstance*>] **width =** <*pixels*> **Height =** <*pixels*>

  [**align =** <*alignment-type*>] [**vspace =** <*pixels*>][**hspace =** <*pixels*>] '>'

  {'<' **param name =** <*Attribute-name*> **value =** <*Attribute-value*> '>'}*

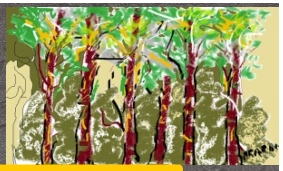
  '</' **Applet** '>'

Example

```
<applet code= gps    height= 200     width= 200>
<param name = font      value = "Arial">
<param name = size     value = "36" >
<param name = style     value = "bold">
</applet>
```

- Applets are treated as a subclass of *java.applet.Applet* class in Java
- Applet subclass inherits many methods from the applet base class
  - URL of the file containing the applet and the applet class definition
  - fetching and rendering the media objects

Security in Web Programming



- Techniques
  - (1) Ensure minimal trust threshold in files and use of digitally signed applets; (2) Safe interpreters; (3) Fault isolation; (4) Code verification.
- Server Trust
  - Severity and the frequency of malicious software transmitted < Threshold
- Safe Interpreter
  - The applet is put in a protected area, and interpreted using a master interpreter instead of executing a compiled code
  - Communication between the applets is regulated to avoid collusion
  - Use of the resources is regulated using limited electronic cash
- Code Verification ensures safe operations
  - Unsafe operations are: (1) 1) operations causing stack overflow or underflow; 2) Namespace violation; 3) Forging of pointers, security managers and class loaders; 4) Local disk accesses; 5) Opening up of its own windows; and 6) communication with processors other than the host or the originating processor
  - .Java loads a security manager to perform code verification
- Fault isolation is done using sandboxing
  - Raising exception when inaccessible address space is accessed