Chapter1

Internet Protocol (IP)

• IP is the fundamental protocol defining the Internet (as the name implies!)
• IP address:
– 32-bit number (in IPv4)
– Associated with at most one device at a time (although device may have more than one)
– Written as four dot-separated bytes, e.g. 192.0.34.166

• IP function: transfer data from source device to destination device
• IP source software creates a packet representing the data
– Header: source and destination IP addresses, length of data, etc.
– Data itself
• If destination is on another LAN, packet is sent to a gateway that connects to more than one network

Transmission Control Protocol (TCP)

• Limitations of IP:
– No guarantee of packet delivery (packets can be dropped)
– Communication is one-way (source to destination)
• TCP adds concept of a connection on top of IP
– Provides guarantee that packets delivered
– Provide two-way (full duplex) communication

• TCP also adds concept of a port
– TCP header contains port number representing an application program on the destination computer
– Some port numbers have standard meanings • Example: port 25 is normally used for email transmitted using the Simple Mail Transfer Protocol (SMTP)
– Other port numbers are available first-comefirst served to any application

User Datagram Protocol (UDP)

• Like TCP in that:
– Builds on IP
– Provides port concept
• Unlike TCP in that:
– No connection concept
– No transmission guarantee
• Advantage of UDP vs. TCP:
– Lightweight, so faster for one-time messages

Domain Name Service (DNS)

• DNS is the “phone book” for the Internet
– Map between host names and IP addresses
– DNS often uses UDP for communication

• Host names

– Labels separated by dots, e.g., www.example.org

– Final label is *top-level domain* • Generic: .com, .org, etc. • Country-code: .us, .il, etc.

• Domains are divided into second-level

domains, which can be further divided into subdomains, etc.

– E.g., in www.example.com, example is a second-level domain

• A host name plus domain name information is called the fully qualified domain name of the computer

– Above, www is the host name, www.example.com is the FQDN

• nslookup program provides command-line access to DNS (on most systems)

• looking up a host name given an IP address is known as a reverse lookup

– Recall that single host may have mutliple IP addresses.

– Address returned is the canonical IP address specified in the DNS system.

Hypertext Transport Protocol (HTTP)

- HTTP is based on the request-response communication model:
  - Client sends a request
  - Server sends a response
- HTTP is a stateless protocol:
  - The protocol does not require the server to remember anything about the client between requests

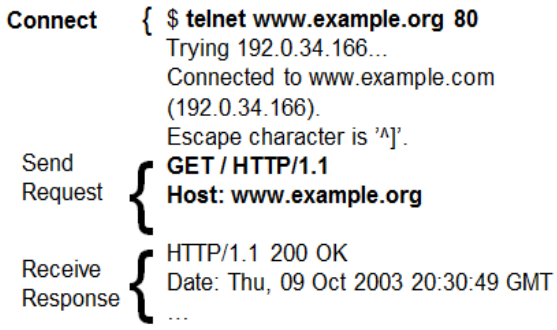
• Normally implemented over a TCP connection

(80 is standard port number for HTTP)

• Typical browser-server interaction: – User enters Web address in browser – Browser uses DNS to locate IP address – Browser opens TCP connection to server – Browser sends HTTP request over connection – Server sends HTTP response to browser over connection – Browser displays body of response in the client area of the browser window

Http:

- The information transmitted using HTTP is often entirely text
- Can use the Internet's Telnet protocol to simulate browser request and view server response



Http request

- Structure of the request:
  - **start line**
  - header field(s)
  - blank line
  - optional body

- **Start line**
  - Example: GET / HTTP/1.1
- Three space-separated parts:
  - HTTP request method
  - Request-URI
  - **HTTP version**
    - We will cover 1.1, in which version part of start line must be exactly as shown

HTTP Request

• Start line
– Example: GET / HTTP/1.1
• Three space-separated parts:
– HTTP request method
– Request-URI
– **HTTP version** • We will cover 1.1, in which version part of start line must be exactly as shown

Uniform Resource Identifier (URI)
– Syntax: *scheme* : *scheme-depend-part* • Ex: In http://www.example.com/ the scheme is http
– Request-URI is the portion of the requested URI that follows the host name (which is supplied by the required Host header field) • Ex: / is Request-URI portio

- **Uniform Resource Identifier (URI)**
  - Syntax: *scheme* : *scheme-depend-part*
    - Ex: In http://www.example.com/
      the scheme is http
  - Request-URI is the portion of the requested URI that follows the host name (which is supplied by the required Host header field)
    - Ex: / is Request-URI portion of http://www.example.com/

URI's are of two types:

- Uniform Resource Name (URN)
  - Can be used to identify resources with unique names, such as books (which have unique ISBN's)
  - Scheme is urn
- Uniform Resource Locator (URL)
  - Specifies location at which a resource can be found
  - In addition to http, some other URL schemes are https, ftp, mailto, and file

Common request methods:

- GET
  - Used if link is clicked or address typed in browser
  - Nobody in request with GET method
- POST
  - Used when submit button is clicked on a form
  - Form information contained in body of request
- HEAD: Requests that only header fields (no body) be returned in the response

Header field structure:

- *field name* : *field value*

- Syntax
  - Field name is not case sensitive
  - Field value may continue on multiple lines by starting continuation lines with white space
  - Field values may contain MIME types, quality values, and wildcard characters (*'s)

Multipurpose Internet Mail Extensions (MIME)

- Convention for specifying content type of a message
  - In HTTP, typically used to specify content type of the body of the response
- MIME content type syntax: -*top-level type* / *subtype*
- Examples: text/html, image/jpeg

HTTP Quality Values and Wildcards

• Example header field with quality values: accept: text/xml,text/html;q=0.9, text/plain;q=0.8, image/jpeg, image/gif;q=0.2,*/*;q=0.1

• Quality value applies to all preceding items
• Higher the value, higher the preference

• Common header fields:

– Host: host name from URL (required)
– User-Agent: type of browser sending request
– Accept: MIME types of acceptable documents
– Connection: value close tells server to close connection after single request/response
– Content-Type: MIME type of (POST) body, normally application/x-www-form-urlencoded
– Content-Length: bytes in body
– Referer: URL of document containing link that supplied URI for this HTTP request

Http response

- Structure of the response:
  - status line
  - header field(s)
  - blank line
  - optional body

- **Status line**
  - Example: HTTP/1.1 200 OK
- Three space-separated parts:
  - HTTP version
  - status code
  - reason phrase (intended for human use)
- Status code
  - Three-digit number
  - First digit is class of the status code:
    - 1=Informational
    - 2=Success
    - 3=Redirection (alternate URL is supplied)
    - 4=Client Error
    - 5=Server Error
  - Other two digits provide additional information

HTTP Response

• Common header fields:
– Connection, Content-Type, Content-Length
– Date: date and time at which response was generated (required)
– Location: alternate URI if status is redirection
– Last-Modified: date and time the requested resource was last modified on the server
– Expires: date and time after which the client's copy of the resource will be out-of-date
– ETag: a unique identifier for this version of the requested resource (changes if resource changes)

Client Caching

- A cache is a local copy of information obtained from some other source
- Most web browsers use cache to store requested resources so that subsequent requests to the same resource will not necessarily require an HTTP request/response
  - Ex: icon appearing multiple times in a Web page

Cache advantages

- (Much) faster than HTTP request/response
- Less network traffic
- Less load on server

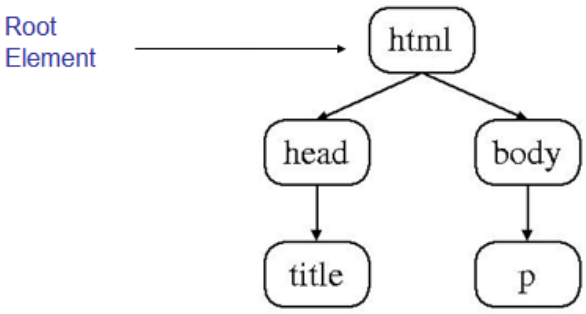
Cache disadvantage

- Cached copy of resource may be invalid (inconsistent with remote version)

Validating cached resource:

- Send HTTP HEAD request and check Last-Modified or ETag header in response
- Compare current date/time with Expires header sent in response containing resource
- If no Expires header was sent, use heuristic algorithm to estimate value for Expires

Chapter2



HTML Tags and Elements

- Any string of the form < … > is a *tag*
- All tags in document instance of Hello World are either end tags (begin with </) or start tags (all others)
  - Tags are an example of markup, that is, text treated specially by the browser
  - Non-markup text is called character data and is normally displayed by the browser
- String at beginning of start/end tag is an element name
- Everything from start tag to matching end tag, including tags, is an element
  - Content of element excludes its start and end tags

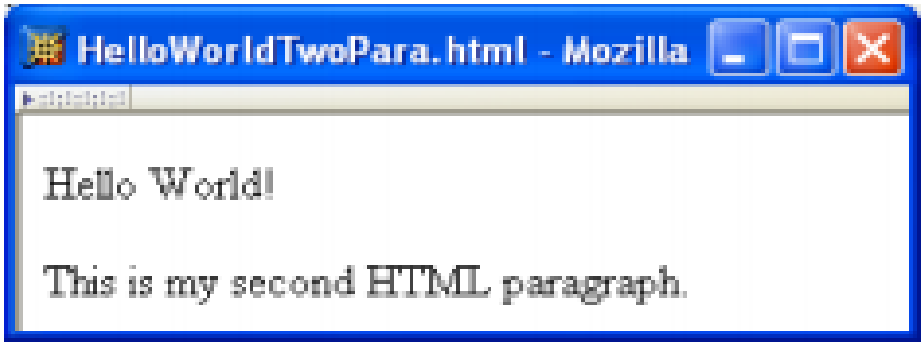
XHTML White Space

• Four white space characters: carriage return, line feed, space, horizontal tab
• Normally, character data is normalized:
– All white space is converted to space characters
– Leading and trailing spaces are trimmed
– Multiple consecutive space characters are replaced by a single space character

```
<body>
  <p>
    Hello World!

    This is my second HTML paragraph.
  </p>
</body>
```

```
<p>
  Hello World!
</p>
<p>
  This is my second HTML paragraph.
</p>
```



Common HTML Elements

- Headings are produced using h1, h2, …, h6 elements:
- Should use h1 for highest level, h2 for next highest, etc.

```
<h1>
  Some Common HTML Elements
</h1>
<h2>
  Simple formatting elements
</h2>
```

- br element represents line break
- br is example of an empty element, i.e., element that is not allowed to have content
- XML allows two syntactic representations of empty elements
  - Empty tag syntax
 is recommended for browser compatibility
  - XML parsers also recognize syntax
</br> (start tag followed immediately by end tag), but many browsers do not understand this for empty elements

Text can be formatted in various ways:

- Apply style sheet technology (next chapter) to a span element (a styleless wrapper):

```
<span style="font-style:italic">separating line</span>
```

- Use a phrase element that specifies semantics of text (not style directly):

```
<strong>hr</strong>
```

- Use a font style element
  - Not recommended, but frequently used

TABLE 2.3: HTML font style elements.

Element	Font used by content
b	Bold-face
i	Italic
tt	"Teletype" (fixed-width font)
big	Increased font size
small	Decreased font size

Images can be embedded using img element

```
<img
    src="http://www.w3.org/Icons/valid-xhtml10"
    alt="Valid XHTML 1.0!" height="31" width="88"
    style="float:right" />
```

- Attributes:
  - src: URL of image file (required). Browser generates a GET request to this URL.
  - alt: text description of image (required)
  - height / width: dimensions of area that image will occupy (recommended)

- Hyperlinks are produced by the anchor element a

```
See
<a href="http://www.w3.org/TR/html4/index/elements.html">the
  W3C HTML 4.01 Element Index</a>
for a complete list of elements.
```

- Anchors can be used as source (previous example) or destination

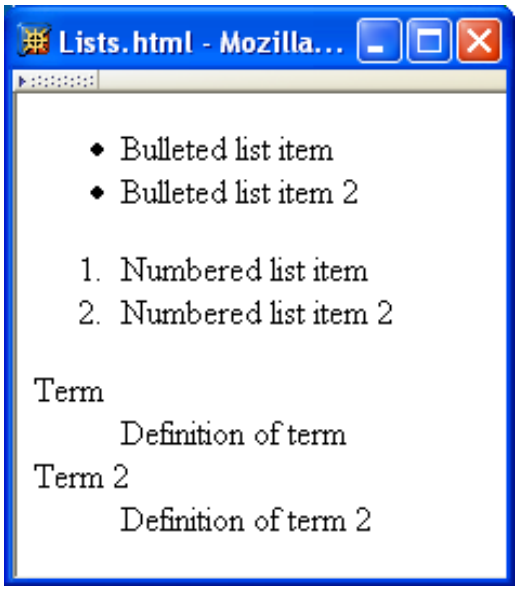
```
<a id="section1" name="section1"></a>
```

- The fragment portion of a URL is used to reference a destination anchor

```
<a href="http://www.example.org/PageWithAnchor.html#section1">...
```

- Browser scrolls so destination anchor is at (or near) top of client area
- Most HTML elements are either block or inline
  - Block: browser automatically generates line breaks before and after the element content
    - Ex: p
  - Inline: element content is added to the "flow"
    - Ex: span, tt, strong, a

List



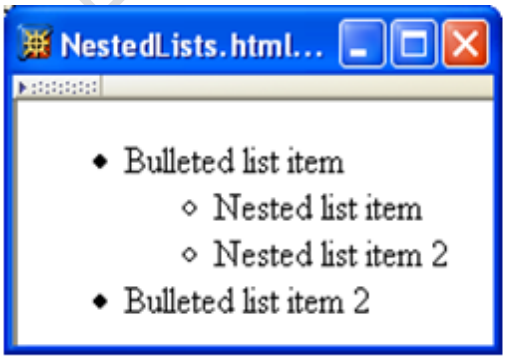
Unordered List

Ordered List

Definition List

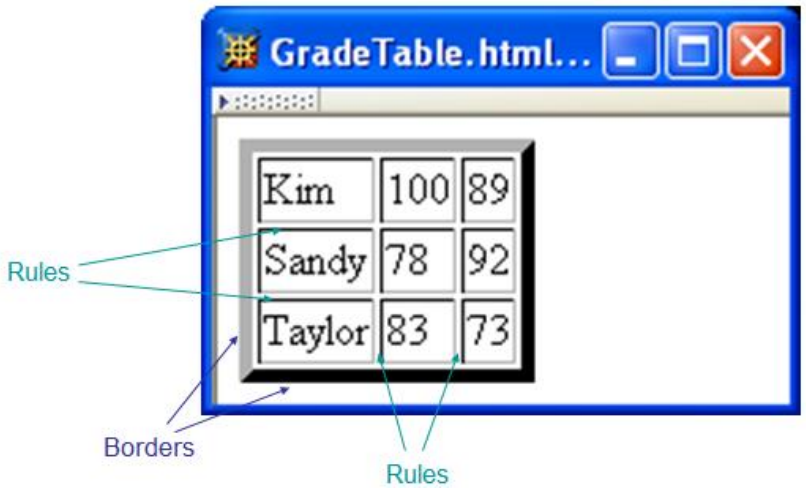
List Items

```
<ul>
  <li>Bulleted list item</li>
  <li>Bulleted list item 2</li>
</ul>
<ol>
  <li>Numbered list item</li>
  <li>Numbered list item 2</li>
</ol>
<dl>
  <dt>Term</dt>
  <dd>Definition of term</dd>
  <dt>Term 2</dt>
  <dd>Definition of term 2</dd>
</dl>
```




```
<ul>
  <li>Bulleted list item
    <ul>
      <li>Nested list item</li>
      <li>Nested list item 2</li>
    </ul>
  </li>
  <li>Bulleted list item 2</li>
</ul>
```

Tables

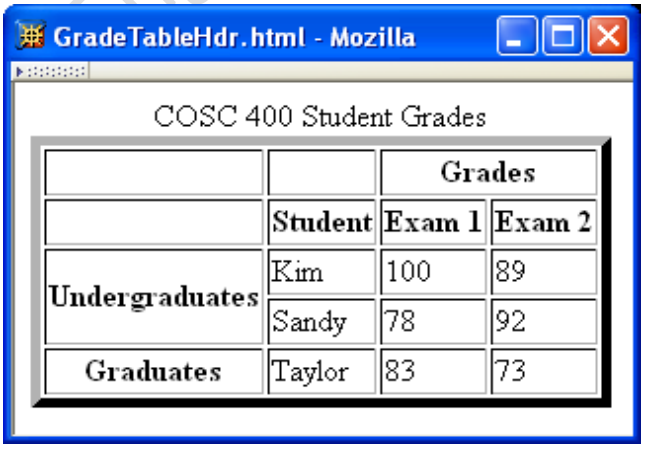


Border 5 pixels, rules 1 pixel

```
<table border="5">
  <tr>
    <td>Kim</td><td>100</td><td>89</td>
  </tr>
  <tr>
    <td>Sandy</td><td>78</td><td>92</td>
  </tr>
  <tr>
    <td>Taylor</td><td>83</td><td>73</td>
  </tr>
</table>
```

Table Row

Table Data



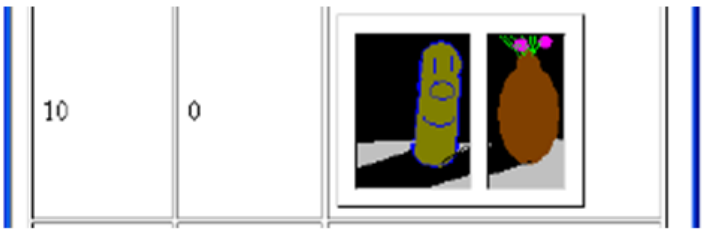
		Grades	
	Student	Exam 1	Exam 2
Undergraduates	Kim	100	89
	Sandy	78	92
Graduates	Taylor	83	73

```
<table border="5">
  <caption>
    COSC 400 Student Grades
  </caption>
  <tr>
    <td> </td><td> </td><th colspan="2">Grades</th>
  </tr>
  <tr>
    <td> </td><th>Student</th><th>Exam 1</th><th>Exam 2</th>
  </tr>
  <tr>
    <th rowspan="2">Undergraduates</th><td>Kim</td><td>100</td><td>89</td>
  </tr>
  <tr>
    <td>Sandy</td><td>78</td><td>92</td>
  </tr>
  <tr>
    <th>Graduates</th><td>Taylor</td><td>83</td><td>73</td>
  </tr>
</table>
```

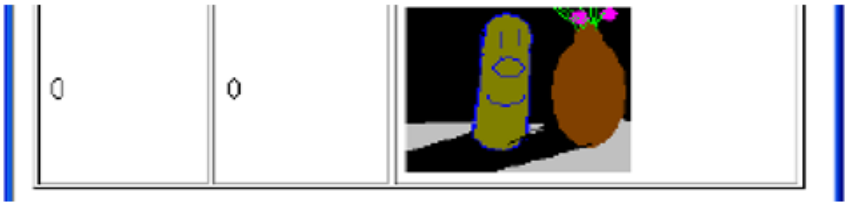
Table Header

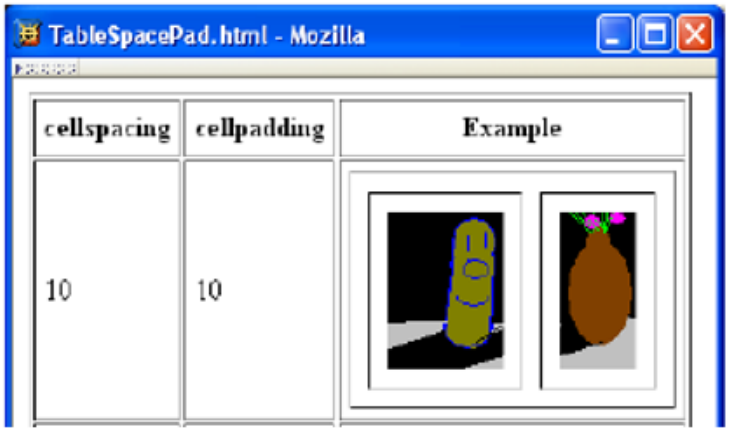
cellspacing cellpadding

cellspacing cellpadding



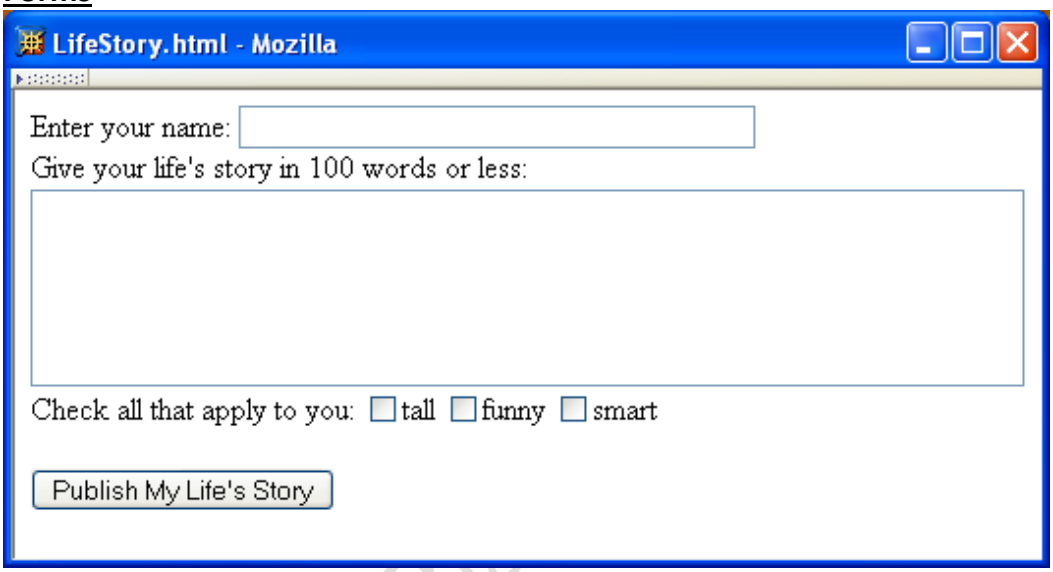
cellspacing cellpadding






```
<table border="1" cellspacing="10" cellpadding="10">
```

Forms



action specifies URL where form data is sent in an HTTP request

Each form is content of a form element

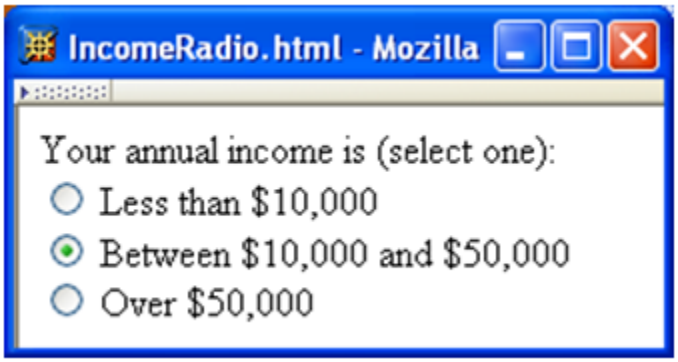
HTTP request method (lower case)

```
<form action="http://www.example.org" method="get">
  <div>
    <label>
      Enter your name: <input type="text" name="username" size="40" />
    </label>
    <br />
    <label>
      Give your life's story in 100 words or less:
      <br />
      <textarea name="lifestory" rows="5" cols="60"></textarea>
    </label>
    <br />
```

Value sent in HTTP request if box is checked

Checkbox control

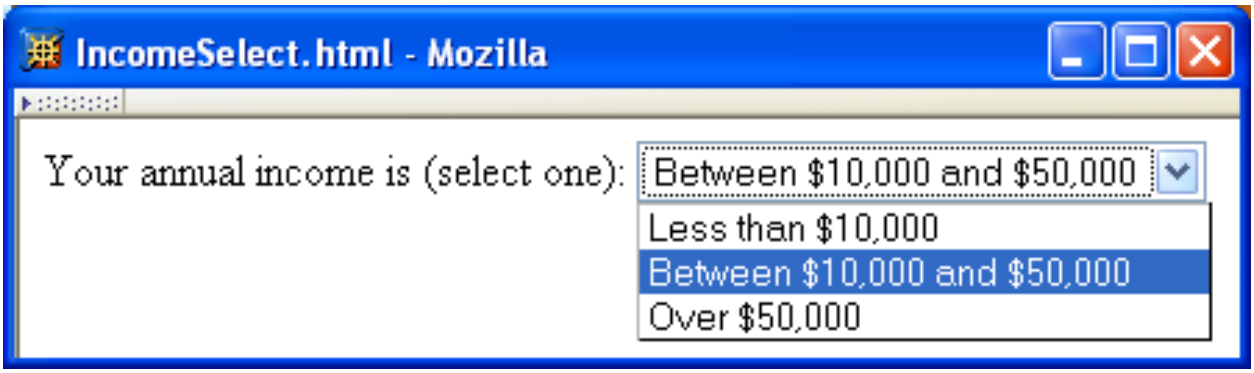
```
Check all that apply to you:
<label>
  <input type="checkbox" name="boxgroup1" value="tall" />tall
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="funny" />funny
</label>
<label>
  <input type="checkbox" name="boxgroup1" value="smart" />smart
</label>
<br /><br />
<input type="submit" name="doit" value="Publish My Life's Story" />
</div>
</form>
```



Radio buttons: at most one can be selected at a time.

```
Your annual income is (select one):<br />
<label>
  <input type="radio" name="radgroup1" value="0-10" />
    Less than $10,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="10-50"
         checked="checked" />
    Between $10,000 and $50,000
</label><br />
<label>
  <input type="radio" name="radgroup1" value="&gt;50" />
    Over $50,000
</label>
```

All radio buttons with the same name form a *button set*




```
Your annual income is (select one):
<select name="income">
  <option value="0-10">Less than $10,000</option>
  <option value="10-50" selected="selected">
    Between $10,000 and $50,000
  </option>
  <option value="&gt;50">Over $50,000</option>
</select>
```

Each menu item has its own value

Item initially displayed in menu control

Chapter3

Cascading Style Sheets (CSS)

- A style sheet technology designed to work with HTML and XML documents.
- CSS provides a great deal of control over the presentation of a document,

Style Sheet Languages

- Cascading Style Sheets (CSS)
  - Applies to (X)HTML as well as XML documents in general
- Extensible Stylesheet Language (XSL)
  - Often used to transform one XML document to another form, but can also add style
  - XSL Transformations covered in later chapter

How to use css

- Style sheets referenced by link HTML element are called external style sheets
- Style sheets can be **embedded** directly in HTML document using style element

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      CSSHelloWorld.html
    </title>
    <link rel="stylesheet" type="text/css" href="style1.css"
          title="Style 1" />
    <link rel="alternate stylesheet" type="text/css" href="style2.css"
          title="Style 2" />
  </head>
  <body>
    <p>
```

```
<head>
  <title>InternalStyleSheet.html</title>
  <style type="text/css">
    h1, h2 { background-color:aqua }
  </style>
</head>
```

CSS Syntax:

Selector Strings

- Single element type:

```
p     { font-size:smaller; letter-spacing:1em }
```

- Multiple element types:

```
h1,h2,h3,h4,h5,h6 { background-color:purple }
```

- All element types:

Universal selector

```
* { font-weight:bold }
```

- Specific elements by id:

```
#p1, #p3 { background-color:aqua }
```

ID selector

- Elements belonging to a style class:
  - Referencing a style class in HTML:

```
#p4, .takeNote { font-style:italic }
```

class selector

- Elements of a certain type and class:

```
span.special { font-size:x-large }
```

this rule applies only to span's belonging to class special

- Source anchor elements:

```
a:link { color:black }
a:visited { color:yellow }
a:hover { color:green }
a:active { color:red }
```

pseudo-classes

- Element types that are descendents:

rule applies to li element that is

```
ul ol li { letter-spacing:1em }
```

rule applies to li element that is
part of the content of an ol element

rule applies to li element that is
part of the content of an ol element
that is part of the content of a ul element

CSS Font Properties

- A font family is a collection of related fonts (typically differ in size, weight, etc.)

```
<p style="font-family:'Jenkins v2.0'">
```

- font-family property can accept a list of families, including generic font families

```
font-family:"Edwardian Script ITC","French Script MT",cursive
```

first choice font — second choice font — generic

Property	Possible Values
font-style	normal (initial value), italic (more cursive than normal), or oblique (more slanted than normal)
font-weight	bold or normal (initial value) are standard values, although other values can be used with font families having multiple gradations of boldness (see CSS2 [W3C-CSS-2.0] for details)
font-variant	small-caps, which displays lowercase characters using uppercase glyphs (small uppercase glyphs if possible), or normal (initial value)

- **font shortcut property:**

```
{ font: italic bold 12pt "Helvetica",sans-serif }
```

```
{ font-style: italic;
  font-variant: normal;
  font-weight: bold;
  font-size: 12pt;
  line-height: normal;
  font-family: "Helvetica",sans-serif }
```

Initial values used if no value specified in font property list

CSS Text Color

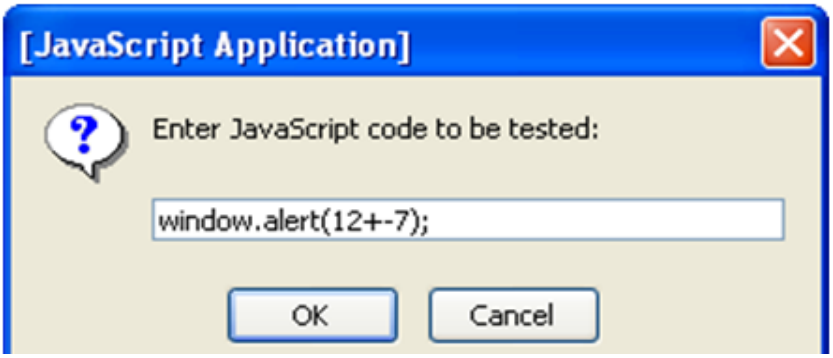
- Font color specified by color property
- Two primary ways of specifying colors:
  - Color name: black, gray, silver, white, red, lime, blue, yellow, aqua, ..etc
  - red/green/blue (RGB) values

Chapter4

JavaScript is a scripting language: designed to be executed within a larger software environment

Prompt window example:

```
var inString = window.prompt("Enter JavaScript code to be tested:",
                             "");
```



Variables and Data Types

- Type of a variable is dynamic: depends on the type of data it contains
- JavaScript has six data types:
  - Number
  - String
  - Boolean (values true and false)
  - Object
  - Null (only value of this type is null)
  - Undefined (value of newly created variable)
- Primitive data types: all but Object

- typeof operator returns string related to data type
  - Syntax: typeof *expression*

Example:

```
// TypeOf.js
var i;
var j;
j = "Not a number";
alert("i is " + (typeof i) + "\n" +
      "j is " + (typeof j));
```

Syntax rules for names (identifiers):

- Must begin with letter or underscore (_)
- Must contain only letters, underscores, and digits (or certain other characters)
- Must not be a reserved word

JavaScript Operators

- Operators are used to create compound expressions from simpler expressions
- Operators can be classified according to the number of operands involved:

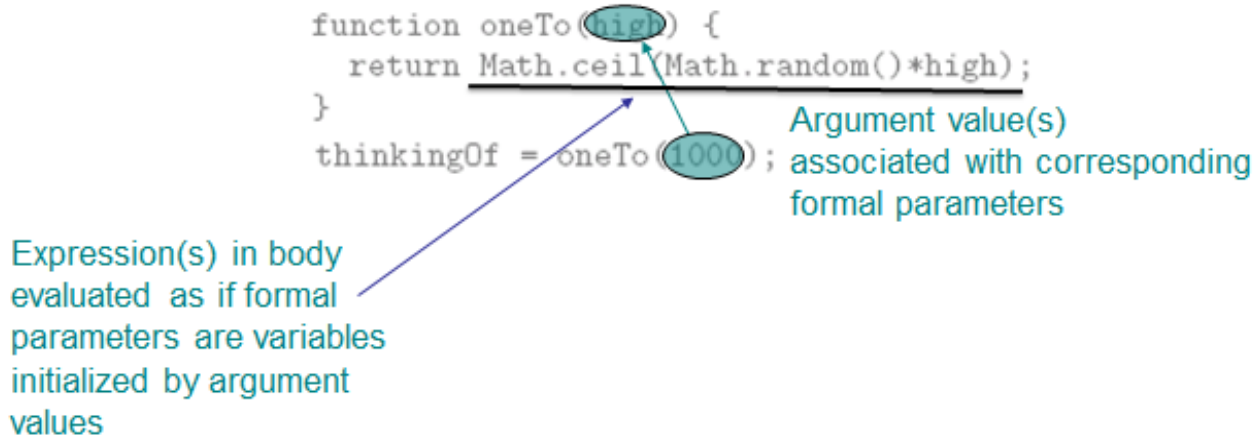
  - Unary: one operand (*e.g.*, typeof i)
    - Prefix or postfix (*e.g.*, ++i or i++)
  - Binary: two operands (*e.g.*, x + y)
  - Ternary: three operands (conditional operator)

Automatic Type Conversion

- Binary operators +, -, *, /, % convert both operands to Number
  - Exception: If one of operands of + is String then the other is converted to String
- Relational operators <, >, <=, >= convert both operands to Number
  - Exception: If both operands are String, no conversion is performed and lexicographic string comparison is performed
- Operators ==, != convert both operands to Number
  - Exception: If both operands are String, no conversion is performed (lex. comparison)
  - Exception: values of Undefined and Null are equal(!)
  - Exception: instance of Date built-in "class" is converted to String (and host object conversion is implementation dependent)
  - Exception: two Objects are equal only if they are references to the same object

- Operators ===, !== are strict:
  - Two operands are === only if they are of the same type and have the same value
  - "Same value" for objects means that the operands are references to the same object
- Unary +, - convert their operand to Number
- Logical &&, ||, ! convert their operands to Boolean (normally)

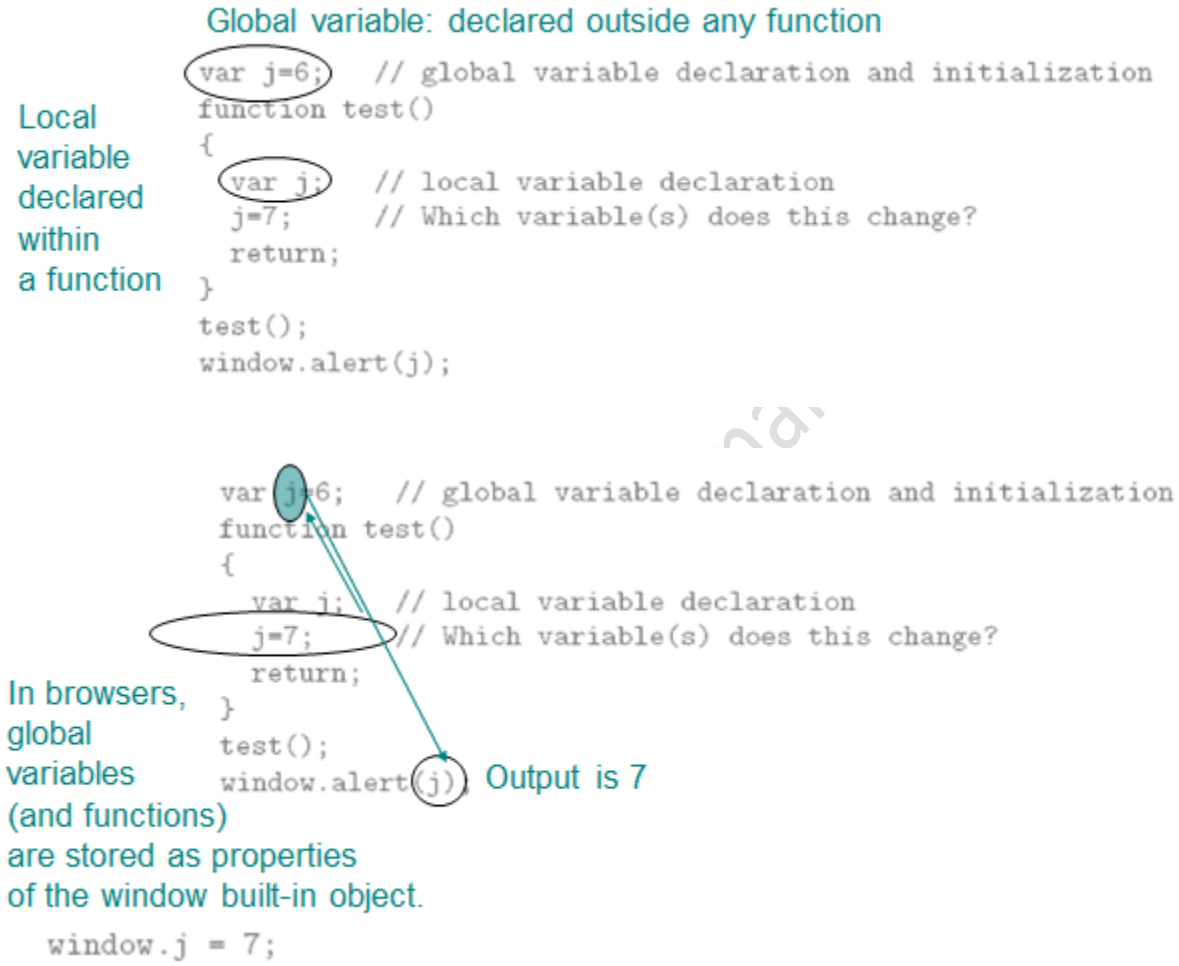
JavaScript Functions



- **Function call semantics details:**
  - Arguments:
    - May be expressions:
    - Object's effectively passed by reference (more later)
  - Formal parameters:
    - May be assigned values, argument is not affected

  - Return value:
    - If last statement executed is not return-value, then returned value is of type Undefined
- Number mismatch between argument list and formal parameter list:
  - More arguments: excess ignored
  - Fewer arguments: remaining parameters are Undefined

Local vs. global variables



- Explicit type conversion supplied by built-in functions
  - Boolean(), String(), Number()
  - Each takes a single argument, returns value representing argument converted according to type-conversion rules given earlier

Object Introduction

- An object is a set of properties
- A property consists of a unique (within an object) name with an associated value
- The type of a property depends on the type of its value and can vary dynamically

```
o.prop = true;      prop is Boolean
o.prop = "true";    prop is now String
o.prop = 1;         prop is now Number
```

Object Creation

- Objects are created using new expression

new Object() *Constructor* and argument list

- A constructor is a function
  - When called via new expression, a new empty Object is created and passed to the constructor along with the argument values
  - Constructor performs initialization on object
    - Can add properties and methods to object
    - Can add object to an inheritance hierarchy

Property Creation

- Assignment to a non-existent (even if inherited) property name creates the property:

```
o1.testing = "This is a test";
```

- Object initializer notation can be used to create an object (using Object() constructor) and one or more properties in a single statement:

```
var o2 = { p1:5+9, p2:null, testing:"This is a test" };
```

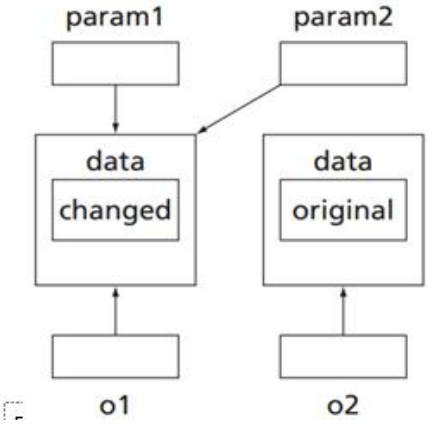

Object argument values are references

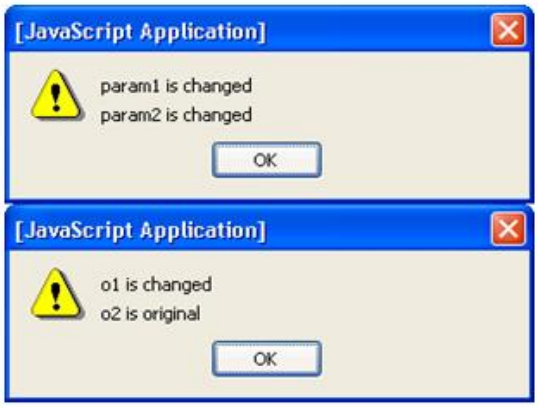
```
function objArgs(param1, param2) {
  // Change the data in param1 and its argument
  param1.data = "changed";
  // Change the object referenced by param2, but not its argument
  param2 = param1;

  window.alert("param1 is " + param1.data + "\n" +
               "param2 is " + param2.data);
  return;
}

// Create two different objects with identical data
var o1 = new Object();
o1.data = "original";
var o2 = new Object();
o2.data = "original";
```

```
// Call the function on these objects and display the results
objArgs(o1, o2);
window.alert("o1 is " + o1.data + "\n" +
             "o2 is " + o2.data);
```





Chapter5

The Document Object Model (DOM) is an API that allows programs to interact with HTML (or XML) documents

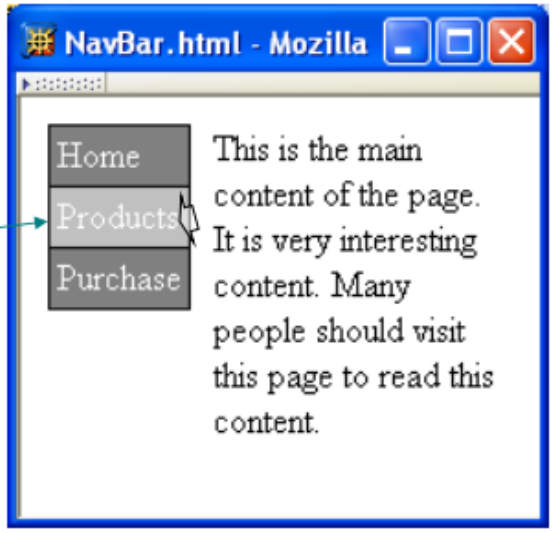
- In typical browsers, the JavaScript version of the API is provided via the document host object

Intrinsic Event Handling

Attribute	When Called
`onload`	The body of the document has just been fully read and parsed by the browser (this attribute pertains only to `body` and `frameset`).
`onunload`	The browser is ready to load a new document in place of the current document (this attribute only pertains to `body` and `frameset`).
`onclick`	A mouse button has been clicked and released over the element.
`ondblclick`	The mouse has been double-clicked over the element.
`onmousedown`	The mouse has been clicked over the element.
`onmouseup`	The mouse has been released over the element.
`onmouseover`	The mouse has just moved over the element.
`onmousemove`	The mouse has moved from one location to another over the element.
`onmouseout`	The mouse has just moved away from the element.

`onblur`	The element has just lost the keyboard focus (attribute pertains only to the same elements as `onfocus`).
`onkeypress`	This element has the focus, and a key has been pressed and released.
`onkeydown`	This element has the focus, and a key has been pressed.
`onkeyup`	This element has the focus, and a key has been released.
`onsubmit`	This element is ready to be submitted (applies only to `form` elements).
`onreset`	This element is ready to be reset (applies only to `form` elements).
`onselect`	Text in this element has been selected (highlighted) in preparation for editing (applies only to `input` and `textarea` elements).
`onchange`	The value of this element has changed (applies only to `input`, `textarea`, and `select` elements).

Modifying Element Style



Change background color of element containing cursor

Like rollover, style needs to be modified both when entering and exiting the element.

Reference to Element instance representing the td element

```
<td onmouseover="highlight(this);"
  onmouseout="lowlight(this);"><a
    href="http://www.example.org"
    >Products</a>
</td>
```

Net effect: "silver" becomes the specified value for CSS background-color property of td element; browser immediately modifies the window.

Reference to Element instance

```
function highlight(element) {
  element.style.backgroundColor = "silver";
  return;
}
```

All Element instances have a style property with an Object value

Properties of style object correspond to CSS style properties of the corresponding HTML element.

```
function lowlight(element) {
    element.style.backgroundColor = "gray";
    return;
}
```

- Rules for forming style property names from names of CSS style properties:
  - If the CSS property name contains no hyphens, then the style object's property name is the same
    - Ex: color ⟶ color
  - Otherwise, all hyphens are removed and the letters that immediately followed hyphens are capitalized
    - Ex: background-color ⟶ backgroundColor

- Alternate syntax:
- Every DOM2-compliant style object has a setProperty() method

```
function highlight(element) {
   element.style.setProperty("background-color", “silver", "");
   return;
}
```

```
function lowlight(element) {
  element.style.setProperty("background-color", "gray", "");
  return;
}
```

CSS property value

CSS property name (unmodified)

Empty string or “important”

• Advantages of setProperty() syntax:

– Makes it clear that a CSS property is being set rather than merely a property of the style object

– Allows CSS property names to be used as-is rather than requiring modification (which can potentially cause confusion)

- Obtaining *specified* CSS property value:

```
if (element.style.backgroundColor == "gray") {
```

- Alternate DOM2 syntax:

```
if (element.style.getPropertyValue("background-color") == "gray") {
```

```
function changeLight(element) {
      if (element.style.backgroundColor == "gray")
     {
          element.style.backgroundColor = "silver";
      }
    else
      {
           element.style.backgroundColor = "gray";
      }
  return;
}
```

Mouse Events

• DOM2 mouse events

– click

– mousedown

– mouseup

– mousemove
– mouseover
– mouseout
• Event instances have additional properties for mouse events

TABLE 5.7 Properties Added to Event Instances Representing DOM2 Mouse Events

Property	Value
`clientX, clientY`	These properties specify the *x* and *y* offsets (in pixels) of the mouse from the upper left corner of the browser client area. Apply to all events.
`screenX, screenY`	These properties specify the *x* and *y* offsets (in pixels) of the mouse from the upper left corner of the display. Apply to all events.
`altKey, ctrlKey, metaKey, shiftKey`	These properties each have a Boolean value indicating whether or not the corresponding keyboard key was depressed at the time this `Event` instance was generated. Apply to all events.
`button`	Which mouse button was depressed: 0=leftmost, 1=second from left, etc. (reversed for left-handed mouse). Applies to click, mousedown, and mouseup events.
`detail`	Number of times the mouse button has been depressed over the same screen location. Applies to click, mousedown, and mouseup events.
`relatedTarget`	If event is mouseover, then `target` is node being entered, and `relatedTarget` is node being exited. If event is mouseout, then `target` is node being exited, and `relatedTarget` is node being entered.

DOM Event Propagation

- Target of event is lowest-level element associated with event
  - Ex: target is the a element if the link is clicked:

<td>click</td>

- However, event listeners associated with ancestors of the target may also be called

Three types of event listeners:

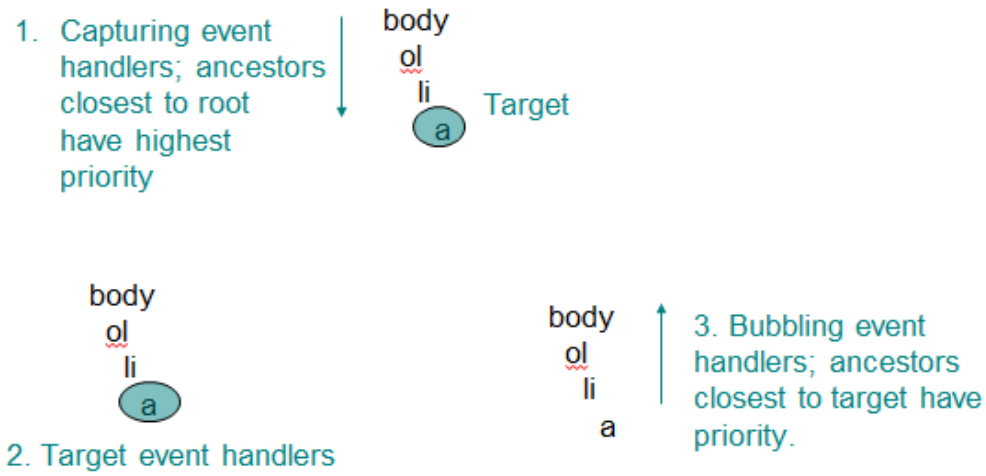
```
<body>
<p id="p1">
  <a id="a1" href="somewhere">Over the rainbow</a>
</p>
<script>
var target = document.getElementById("a1");
var ancestor = document.getElementById("p1");
ancestor.addEventListener("click", listener1, true);
target.addEventListener("click", listener2, false);
ancestor.addEventListener("click", listener3, false);
</script>
</body>
```

Capturing: Listener on ancestor created with true as third arg.

Bubbling: Listener on ancestor created with false as third arg.

Target: Listener on target element

Priority of event handlers:



Certain events do not bubble, *e.g.*,
– load
– unload
– focus
– blur

Propagation-related properties of Event instances:

- eventPhase: represents event processing phase:
  - 1: capturing
  - 2: target
  - 3: bubbling

- currentTarget: object (ancestor or target) associated with this event handler
- Propagation-related method of Event instances:
  - stopPropagation(): lower priority event handlers will not be called
- Typical design:
  - Use bubbling event handlers to provide default processing (may be stopped)
  - Use capturing event handlers to provide required processing (*e.g.*, cursor trail)