



Final Exam Review

INFO/CSE 100, Spring 2005
Fluency in Information Technology

<http://www.cs.washington.edu/100>



Readings and References

- Reading
 - » *Fluency with Information Technology*
 - Chapters 1 - 21
- Labs
 - » Labs 1-9
- References
 - » Various web-based references



Basic Terminology

- Hardware:
 - » Be able to name the different parts of the computer!
 - Monitor, display, Cathode Ray Tube (CRT), Liquid Crystal Display (LCD), pixel, motherboard, daughterboard, processor, RAM, hard disk, mouse
 - » Memory
 - Random Access Memory (RAM), hard disk, kilo (1 thousand), mega (1 million), giga (1 billion)



Basic Terminology

- Software
 - » Operating System, program, algorithm, Graphical User Interface (GUI), command line, boot
- Software Operations
 - » Basic Metaphors
 - Buttons, sliders, close boxes, menus, keyboard shortcuts, ellipses on a menu means “more input required”
 - » Common Operations
 - New, Open, Close, Save, Save as, Print, Print preview, Exit or Quit
 - » Common Editing Operations
 - Cut, Copy, Paste, Clear, Select all, Undo, Repeat



Basic Terminology

- Networking
 - » Types of networks
 - The Internet, Wide Area Network (WAN), Local Area Network (LAN)
 - » Protocols
 - Transmission Control Protocol/Internet Protocol (TCP/IP), File Transfer Protocol (FTP), Ethernet Protocol, Hypertext Transport Protocol (HTTP)
 - » How to Read a Domain Name!
 - Network addresses, IP addresses, domain names, Domain Name Service (DNS)



Comparing Changes

- How fast is the Chevy corvette?!?
 - » 1977 0-60mph in 7.2 secs
 - » 1997 0-60mph in 4.8 secs
- Percent improvement
 - » $\text{new rate} - \text{old rate} / \text{old rate} = \% \text{ improvement}$
 - » $4.8 - 7.2 / 7.2 = .33 = 33\% \text{ improvement}$
- Factor of improvement
 - » $\text{new rate} / \text{old rate} = \text{factor of improvement}$
 - » $4.8/7.2 = .66 \text{ factor of improvement}$



Unix Commands

- `cd dir` - change directory
- `ls` - list directory
- `pwd` - print working directory
- `mkdir dir` - make a new directory
- `pico filename` - open file with pico editor
- `more filename` - read file
- `cp source dest` - copy the source to destination
- `mv source dest` - move the source to destination
- `chmod` - change mode (permissions)
- `rm filename` - remove file
- `rmdir dir` - remove directory (empty)
- `exit / logout` - log out of the remote computer



Pathnames

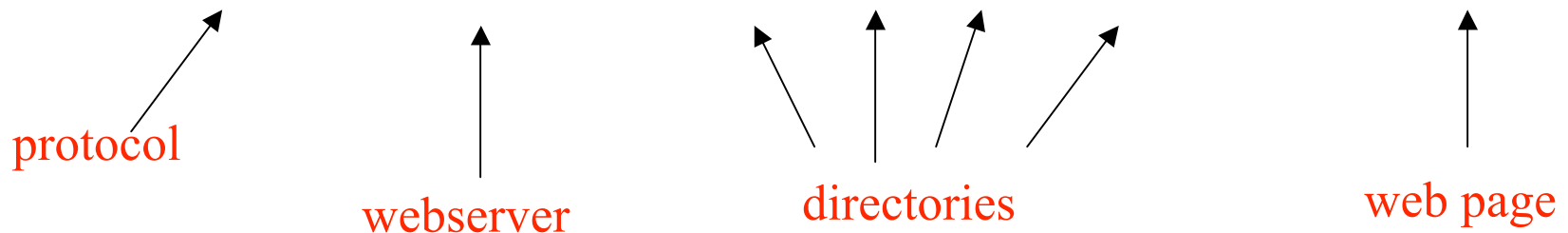
- “root” of a files system is specified with a single “/” slash (or C:\ for Windows OS)
 - » Absolute pathnames start from the root
 - » Relative pathnames start from the current directory
- A single “/” slash is used to separate directories and filenames on Unix (“\” backslash on Windows)
- Know that “..” means the parent directory and “.” means the current directory

```
$pwd
$/suzka/fit100/project1
$cd ../..
$pwd
$/suzka
```


World Wide Web

- Understand how servers and clients (web browsers) interact using HTTP
- Default web pages (index.html)
- Elements of a URL (uniform resource locator)

» <http://www.cnn.com/2005/TECH/04/15/laser.warn/index.html>





HyperText Markup Language

- The language in which web pages are written
- The filename extension is generally .html or .htm
- Plain text with a special structure defined by a set of tags
- Tags are used to encode structure and formatting



HTML Page Structure

```
<!DOCTYPE definition... >
<html>
  <head>
    <title>Title text</title>
  </head>
  <body>
    Body text goes here...
  </body>
</html>
```



HTML Tags

- Anatomy of an HTML tag
 - » `<ELEMENT attribute="value"></ELEMENT>`
 - » ``
- Types of tags
 - » Normally has an open AND a close element
 - » Open and close at the same time
 - » Some tags do not close at all (ex. Comment tag)
- Attributes
 - » Name-value pairs, values in quotes
 - » Some are required, some optional

More HTML Tags

- Styles Tags
 - » `` bold, `<i>` italic, `<u>` underline
- Spacing Tags
 - » `<p>` paragraph, `
` line break, `<hr>` horizontal rule
- Heading tags
 - » `<h1>`, `<h2>`, `<h3>` header format
- Table tags
 - » `<table>` table, `<tr>` table row, `<td>` table data
- References
 - » `` anchor reference
 - » `` image reference



Tips for Debugging

- Verify that its reproducible!!!
- Determine exactly what the problem is
- Eliminate obvious causes by double-checking
- Divide the process into smaller working parts
- Use tools to help you program (like colored text editor)
- Use techniques to help you program (like indenting, adding comments, etc...)



Searching the Web

- Search Engines like Google & Yahoo gives us access to large piles of (unorganized) information
- Indexes are generated by crawling the web and following all the links and indexing words
- Not every page can be indexed!
 - » No other pages link to it
 - » It's a dynamically created page



Search Specifics

- Be as specific as you can when searching the web!
 - » Eliminate common words (a, the, but)
 - » Use rare words
 - » Try using longer queries
 - » Don't forget about advanced search
- Employ Boolean operators
 - » AND = both words must be included (any order)
 - » OR = one or the other word (but not both)
 - » AND NOT = do not include this word
 - » “quotes” to guarantee word order

Information Representation

- Digitization: representing information with a fixed set of symbols
 - » Using positional notation and a fixed set of symbols, any number of states can be identified
- Different encodings can be used to represent the same set of states
- Any phenomenon that can be set and measured can be used to encode state information
- Most common encoding is the PandA (presence and absence)



Bits and Bytes

- A bit is a contraction of “binary digit”
 - » A bit represents one state (like true or false)
 - » A byte is 8 bits
- 256 characters can be encoded in 8-bits because $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$
- Bytes are used to encode characters
 - » Each value is interpreted as a different character code
 - » 0010 1010 <-- 1 byte



Representing Data as Symbols

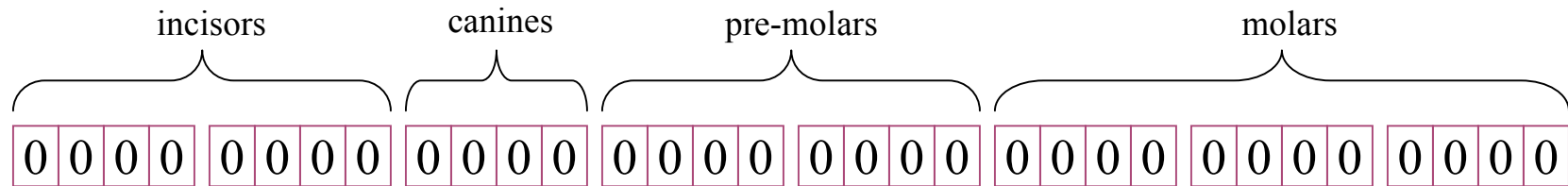
- 24 Greek Letters
- And we decide to use 2 symbols, binary, to represent the data.
- How many bits do we need?!?
 - » 24 total possibilities
 - » $2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$
 - We get 6 extra!

Info Representation

- Adult humans have 32 teeth
 - » sometimes a tooth or two is missing!
- How can we represent a **set** of teeth?
 - » How many different items of information?
 - 2 items - *tooth* or *no tooth*
 - » How many "digits" or positions to use?
 - 32 positions - one per tooth socket
 - » Choose a set of symbols
 - no tooth: 0* *tooth: 1*



What's your tooth number?



no teeth \leftrightarrow 0000 0000 0000 0000 0000 0000 0000 0000



no molars \leftrightarrow 1111 1111 1111 1111 1111 0000 0000 0000

How many possible **combinations**? $2 \times 2 \times 2 \times 2 \times \dots \times 2 = 2^{32} \approx 4$ Billion



How many positions should we use?

It depends: how many numbers do we need?

one
position

0
1

} two numbers

two
positions

0	0
0	1
1	0
1	1

} four numbers

three
positions

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

} eight numbers

Converting from binary to decimal

$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$ <small>$2 \times 2 \times 2 \times 2$</small>	$2^3 = 8$ <small>$2 \times 2 \times 2$</small>	$2^2 = 4$ <small>2×2</small>	$2^1 = 2$ <small>2</small>	$2^0 = 1$ <small>1</small>	base 10
1	0	0	0	1	0	1	0	base 2

$$1 \cdot 128 + 0 \cdot 64 + 0 \cdot 32 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 138_{10}$$

$$1 \cdot 128 + 1 \cdot 8 + 1 \cdot 2 = 138_{10}$$

Each position represents one more multiplication by the base value.

For binary numbers, the base value is 2, so each new column represents a multiplication by 2.

Base 16 Hexadecimal

- The base value can be **16** - *hexadecimal numbers*
 - » Sixteen symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
 - » Each column represents a multiplication by sixteen
 - » Hex is easier to use than binary because the numbers are shorter even though *they represent the same value*

$$\begin{array}{cccc}
 16 \times 16 \times 16 & 16 \times 16 & 16 & 1 \\
 16^3 = 4096 & 16^2 = 256 & 16^1 = 16 & 16^0 = 1
 \end{array}$$

base 10

0	0	8	A
---	---	---	---

base 16

$$8 \cdot 16 + 10 \cdot 1 = 138_{10}$$



Four binary bits \Leftrightarrow One hex digit

binary base 2	hexadecimal base 16	decimal base 10
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7

\Leftrightarrow

\Leftrightarrow

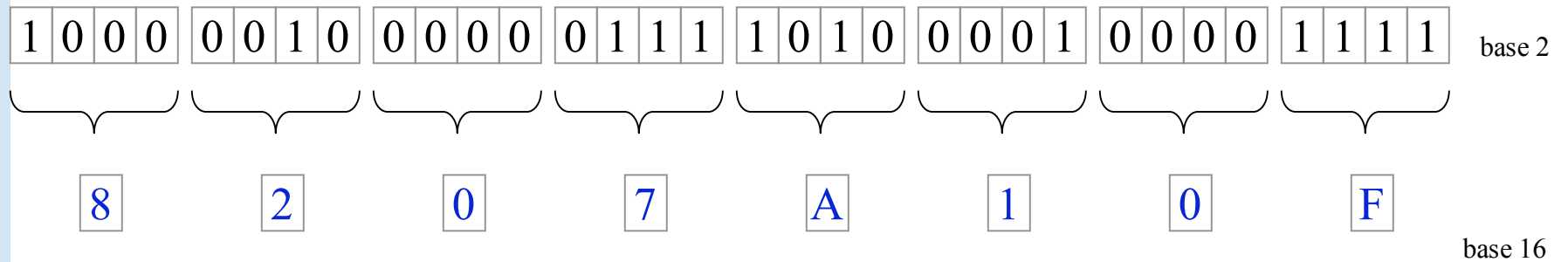
binary base 2	hexadecimal base 16	decimal base 10
1 0 0 0	8	8
1 0 0 1	9	9
1 0 1 0	A	10
1 0 1 1	B	11
1 1 0 0	C	12
1 1 0 1	D	13
1 1 1 0	E	14
1 1 1 1	F	15

\Leftrightarrow

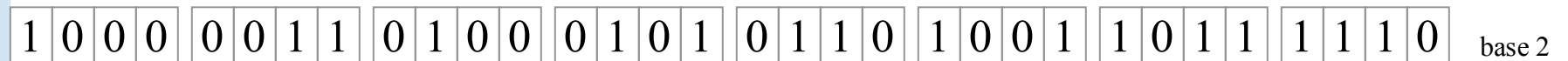
\Leftrightarrow



Binary to Hex examples



$$100000100000001111010000100001111_2 = 8207A10F_{16}$$



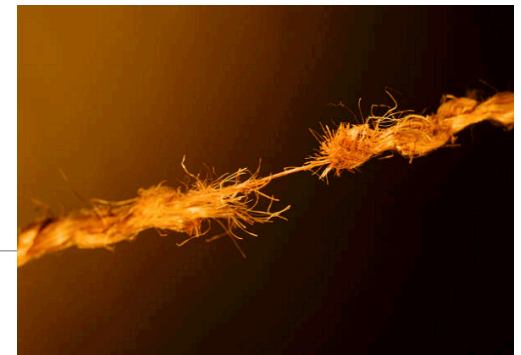
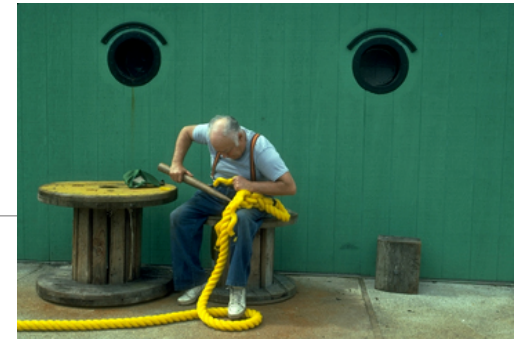
$$10000011010001010110100110111110_2 = \text{_____}_{16}$$

Overview

- During this quarter, we're looking at the actual workings of computer systems
- Organized as “*layers of abstraction*”
 - » application programs
 - » higher level languages: Javascript, SQL, ...
 - » operating system concepts
 - » bits, bytes, assembly language
 - » transistors, electrons, photons

Layers of Abstraction

- At any level of abstraction, there are
 - » elements at that level
 - » the building blocks for those elements
- Abstraction
 - » isolates a layer from changes in the layer below
 - » improves developer productivity by reducing detail needed to accomplish a task
 - » helps define a single architecture that can be implemented with more than one organization





Architecture & Organization

- Architecture (the *logical definition*)
 - » defines elements and interfaces between layers
 - » Instruction Set Architecture
 - instructions, registers, addressing
- Organization (the *physical implementation*)
 - » components and connections
 - » how instructions are implemented in hardware
 - » many different organizations can implement a single architecture



Computer Architecture

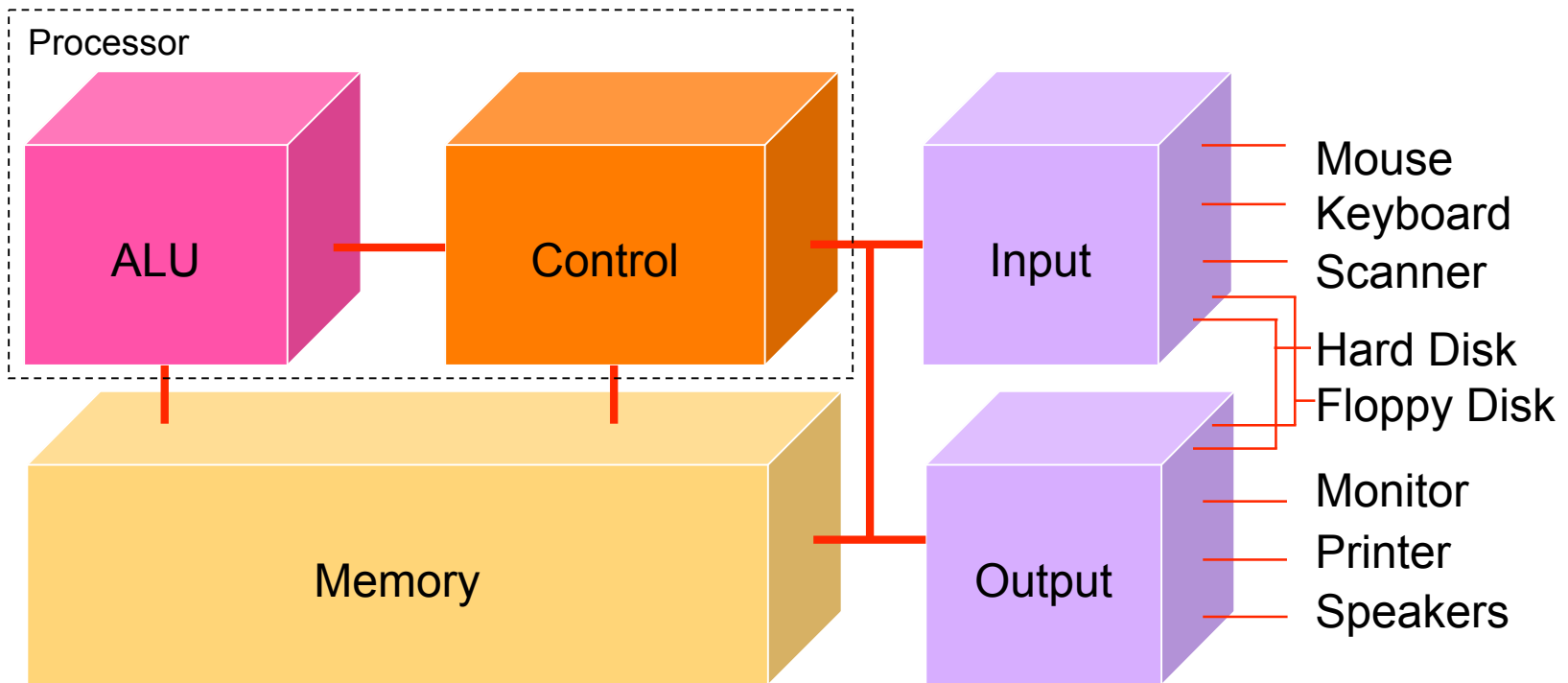
- Specification of how to program a specific computer family
 - » what instructions are available?
 - » how are the instructions formatted into bits?
 - » how many registers and what is their function?
 - » how is memory addressed?
- Some examples architectures
 - » IBM 360, 370, ...
 - » PowerPC 601, 603, G5, ...
 - » Intel x86 286, 386, 486, Pentium, ...
 - » MIPS R2000, R3000, R4000, R5000, ...

Computer Organization

- Processor
 - » Data path (ALU) manipulate the bits
 - » The control controls the manipulation
- Memory
 - » cache memory - smaller, higher speed
 - » main memory - larger, slower speed
- Input / Output
 - » interface to the rest of the world



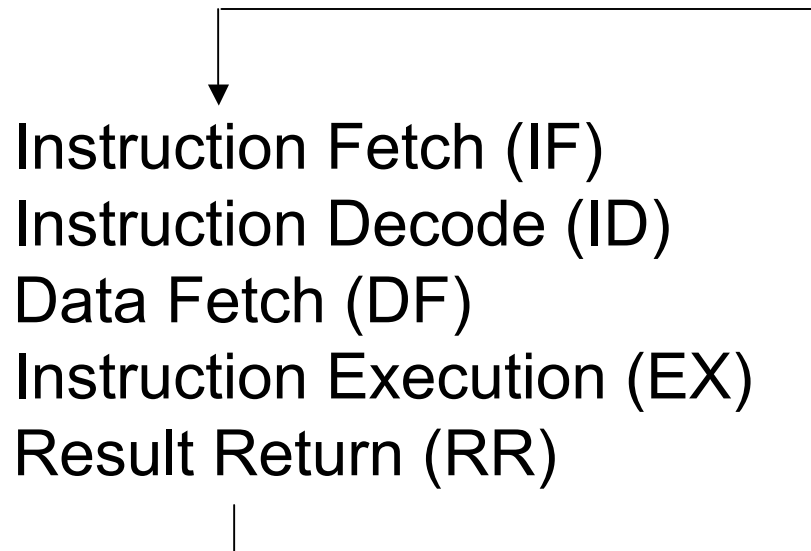
Anatomy of a Computer



Fetch/Execute Cycle

Computer = instruction execution engine

- » The fetch/execute cycle is the process that executes instructions

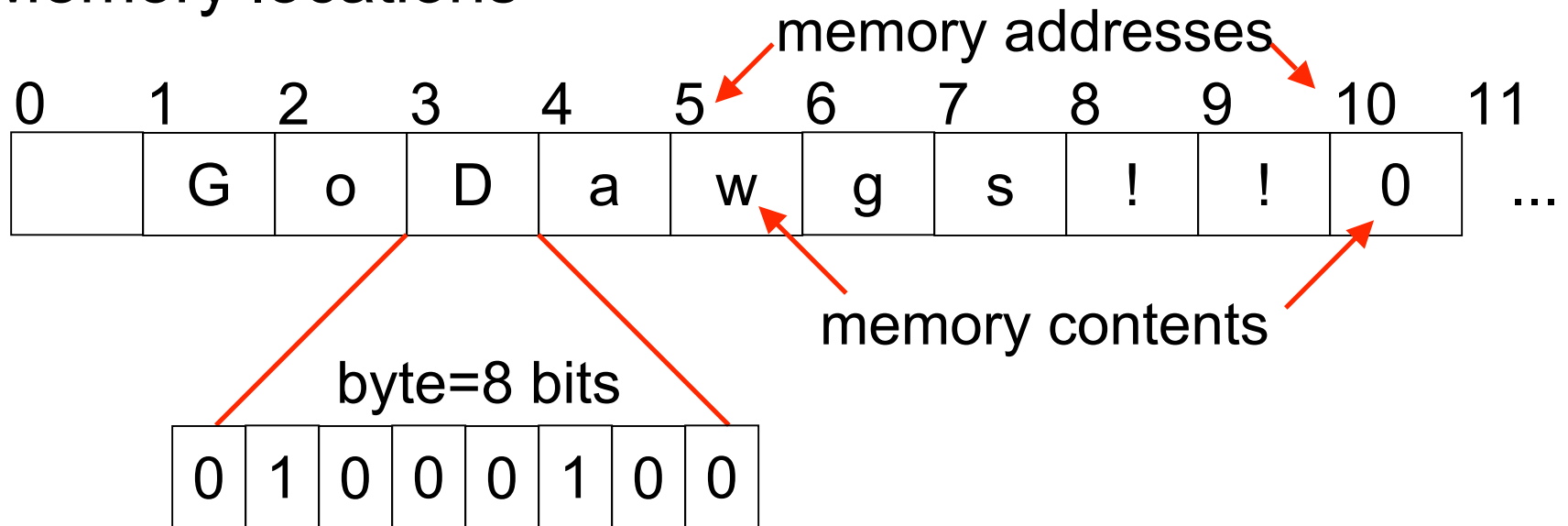




Memory ...

Programs and the data they operate on must be in the memory while they are running

Memory locations



Control

- The Fetch/Execute cycle is hardwired into the computer's control, i.e. it is the actual “engine”
- Depending on the Instruction Set Architecture, the instructions say things like
 - » Put in memory location 20 the contents of memory location 10 + contents of memory location 16
 - » The instructions executed have the form `ADDB 10, 16, 20`
 - Add the bytes from memory address 10 and memory address 16 and store the result in memory address 20

10	11	12	13	14	15	16	17	18	19	20	21
6						12				18	...



ALU

The Arithmetic/Logic Unit does the actual computation

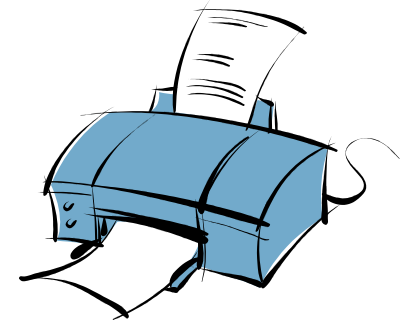
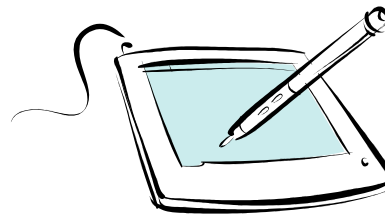
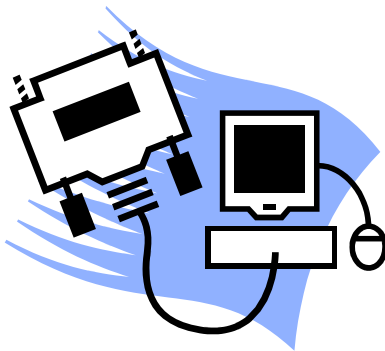
Depending on the Instruction Set Architecture, each type of data has its own separate instructions

ADDDB	: add bytes	ADDDBU	: add bytes unsigned
ADDH	: add half words	ADDHU	: add halves unsigned
ADD	: add words	ADDU	: add words unsigned
ADDS	: add short decimal numbers		
ADDD	: add long decimal numbers		

Most computers have only about a 100-150 instructions hard wired

Input/Output

- Input units bring data to memory from outside world; output units send data to outside world from memory
 - » Most peripheral devices are “dumb”, meaning that the processor assists in their operation



The PC's PC

- The program counter (PC) tells where the next instruction comes from
 - » In some architectures, instructions are always 4 bytes long, so add 4 to the PC to find the next instruction

Program Counter: 112



Clocks Run The Engine

- The rate that a computer “spins around” the Fetch/Execute cycle is controlled by its clock
 - » Current clocks run 2-3 GHz
 - » The computer tries do at least one instruction per cycle, depending on the instruction and the availability of memory contents
 - » Modern processors often try to do more than one instruction per cycle

Clock rate is not a good indicator of speed anymore, because several things are happening every clock cycle



Programs vs Algorithms

- An algorithm is a precise, systematic method to produce a desired result
 - » an Algorithm

```
longStringWithShortStringInIt ← placeholder
ShortString ← e
placeholder ← longStringWithShortStringInIt
```
- A program is an algorithm specialized to a particular situation
 - » a Program that implements the Algorithm

```
↵↵ ← # // replace double <newlines> with <#>
↵ ← e // delete all single <newlines>
# ← ↵↵ // restore all double <newlines>
```


Variables In Programming

- Program variables have names and values
 - » Names (also called identifiers)
 - generally start with a letter and can contain letters, numbers, and underscore characters “_”
 - Names are *case sensitive*
 - » Values
 - can be numbers, strings, Boolean, etc
 - change as the program executes

Variable Name	Current Value	Previous Value
is2005	TRUE	FALSE
ALChampion	Boston Red Sox	New York Yankees
No_1_Box_Office	Shark Tale	Shark Tale
dayOfTheWeek	Monday	Sunday
huskyCardBalance	\$52	\$60

Variable Declarations in Javascript

```
<script type="text/javascript">  
  
var eyeColor; <<< undefined!  
  
var eyeColor = "green"; <<< initialized  
  
var eyeColor = ""; <<< initilized, empty  
  
var eyeColor = "green", hairColor="blonde";  
hairColor = "carmel";  
  
</script>
```



Basic Data Types in Javascript

Numbers:

```
var gasPrice = 2.55;
```

Strings

```
var eyeColor = "hazel green";
```

Boolean

```
var isFriday = true;  
var isWeekend = 0;
```



Expressions

- The right-hand side of an assignment statement can be any valid *expression*
- Expressions are “formulas” saying how to manipulate existing values to compute new values

```
balance = balance - transaction;  
seconds = 60*minutes;  
message = "Status code is " + codeValue;  
isFreezing = (temp < 32);
```



Operators

Use operators to build expressions

- » Numeric operators

 - + - * / *mean* add, subtract, multiply, divide

- $3 + 3 = 6$

- » String operator

 - + *means* concatenate strings

- "3" + "3" = "33"

- » Relational operators

 - < <= == != >= > *mean* less than, less than or equal to, equal to, not equal to, greater than or equal to, greater than

- » Boolean operators

 - && || ! *mean* and, or, not

Functions

A *function* is a way to bundle a set of instructions and give them a name so that you can reuse them easily

Functions have a specific layout

- » *<name>* ← the function name is an identifier
- » *<parameter list>* ← list of input variables for the function
- » *<statements>* ← the statements do the work

```
function <name> ( <parameter list> ) {  
    <statements>  
}
```

Example Function

template

```
function <name> ( <parameter list> ) {  
    <statements>  
}
```

Write a simple function to compute the Body Mass Index when the inputs are in English units (ie, US units)

example

```
// Calculate Body Mass Index in English units  
// weight in pounds  
// height in inches  
// returns body mass index  
  
function bmiE(weightLBS, heightIN) {  
    var heightFt = heightIn / 12; // convert to feet  
    return 4.89 * weightLBS / (heightFt * heightFt);  
}
```

Calling a Function

```
// Calculate Body Mass Index in English units
// weight in pounds
// height in inches
// returns body mass index

function bmiE(weightLBS, heightIN) {
  var heightFt = heightIn / 12; // convert to feet
  return 4.89 * weightLBS / (heightFt * heightFt);
}
```

parameters

function calls

```
// call the bmiE function
var bmi = bmiE(162, 51);

// another function call
document.write(bmiE(162, 51));
```

arguments

Global or Local?!?

- Scope of a variable describes where and when it can be referenced
 - » Local variables are only known inside of a function (curly braces)
 - » Global variables are known by all the Javascript inside of `<script>`
`</script>` pairs

```
// Calculate Percentage of Study Hours/Week
// time in hours
// returns hours
var days = 7;
function calculateStudyHrs(time) {
    var totalHrs = 24 * days;
    return time/totalHrs;
}
```



Layout of the GUI

- The layout of the page is controlled with HTML in the body of the page

```
<body>
```

```
  HTML form layout and specification
```

```
</body>
```

```
</html>
```

- The layout and controls are provided using new tags
 - » `<form id="buttonForm">`
 - » `<button type="button" ...`
 - » `<input type="text" ...`
 - » `<input type="radio" ...`
 - » `<button type="reset" ...`

A simple example

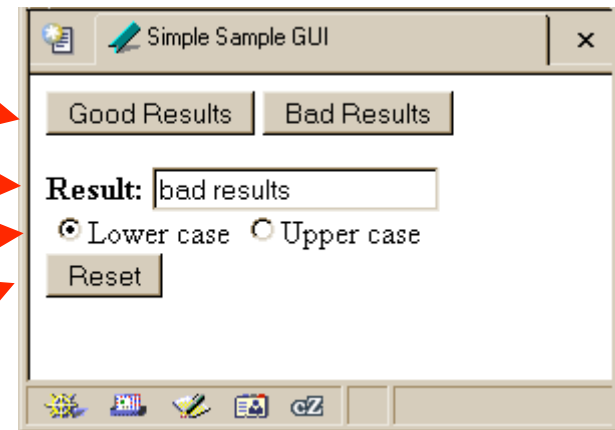
This GUI has several simple controls.

Two buttons to control the results

One text field to display the results

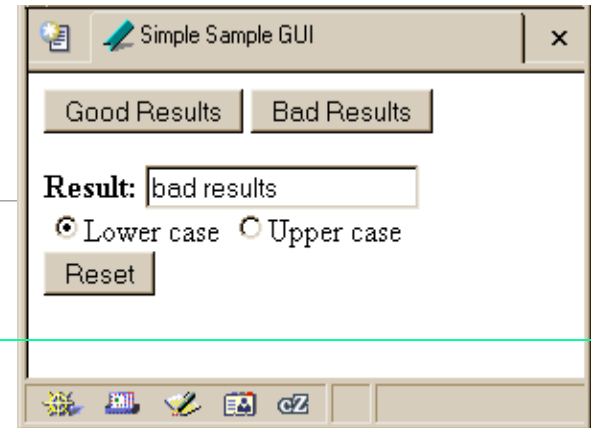
One pair of radio buttons to control the display

One button to reinitialize



<http://www.cs.washington.edu/education/courses/100/04au/slides/13-gui/gui.html>

Form Controls & Events



```

<form>
<button type="button"
  onclick="setResults('good results')">Good Results</button>
<button type="button"
  onclick="setResults('bad results')">Bad Results</button>
<b>Result:</b>
<input type="text" value="nada" readonly id="resultField">
<br>
<input type="radio" name="case" id="radioLC" checked
  onclick="setResults(document.getElementById('resultField').value)
">Lowercase
<input type="radio" name="case" id="radioUC"
  onclick="setResults(document.getElementById('resultField').value)
">Uppercase
<br><button type="reset">Reset</button>
</form>

```

Events Cause Processing

- After drawing a page, the browser sits idle waiting for something to happen ... when we give input, we cause *events*
- Processing events is the task of a block of code called an **event handler**
 - » The code to execute is identified in the tag using the appropriate attribute
 - » There are many event types
 - `onClick`, `onChange`, `onMouseOver` ...



setResults(resultString)

```
<script type="text/javascript">  
function setResults(resultString) {  
    var tempString = resultString;  
    if (document.getElementById("radioLC").checked) {  
        tempString = tempString.toLowerCase();  
    } else if (document.getElementById("radioUC").checked) {  
        tempString = tempString.toUpperCase();  
    }  
    document.getElementById("resultField").value = tempString;  
}  
</script>
```

parameter variable, local variable, if/else statement, field reference,
call to toLowerCase() function

The `if / else` statement

The `if` statement is a *conditional statement*

- » a conditional expression is evaluated as being `true` or `false`
 - the expression is a *boolean expression* (ie, returns `true` or `false`)
- » if the condition is `true`, then one set of statements is executed
- » if the statement is `false`, then a different set of statements is executed

condition
↙

```
if (<boolean expression>) {  
    <statements>  
} else {  
    <statements>  
}
```



Examples

```
if (count == 0) {  
    ready = false;  
} else {  
    ready = true;  
    count = count-1;  
}
```

What is the conditional expression?

What statements are part of the true block?

Which statements are part of the false block?

What happens when count is 21? 0? -1?

```
if (pageCount >= 100) {  
    alert("This may take a few minutes.");  
}
```

What is the conditional expression?

Which statements are part of the false block?

What statements are part of the true block?

What happens when pageCount is 21? 100? 200?

More if/else Statements

```
if (temp < 32) {  
    if (sky == "cloudy") {  
        alert("Snow is forecast!");  
    }  
}
```

```
if (temp < 32 && sky == "cloudy") {  
    alert("Snow is forecast!");  
}
```



The **for** loop

A counting loop is usually implemented with **for**

```
var count = 10;
```

initialize

check for limit

update loop control index
shorthand for **i=i+1**

```
for (var i=0; i < count; i++) {  
    document.writeln("<br>index value is : "+i);  
}
```

one or more statements in the loop body



`i++` is a shortcut

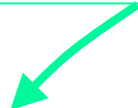
- `for (i=0; i < count; i++)`
- at the end of every pass through the **for** loop body, do the following:
 - » get the value of `i`
 - » increment `i`
 - » store the incremented value
- Used as it is here, this is the same as writing
 - » `i = i + 1`

body of loop may not execute at all

- Notice that depending on the values of the control variables, it is quite possible that the body of the loop will not execute at all

check for limit condition
`itemCount` is 0 when we get here, so
`i < itemCount` is immediately false and
the loop body is skipped completely

```
var itemCount = 0;
...
for (var i=0; i < itemCount; i++) {
    document.writeln("<br>..processing item "+i);
}
```

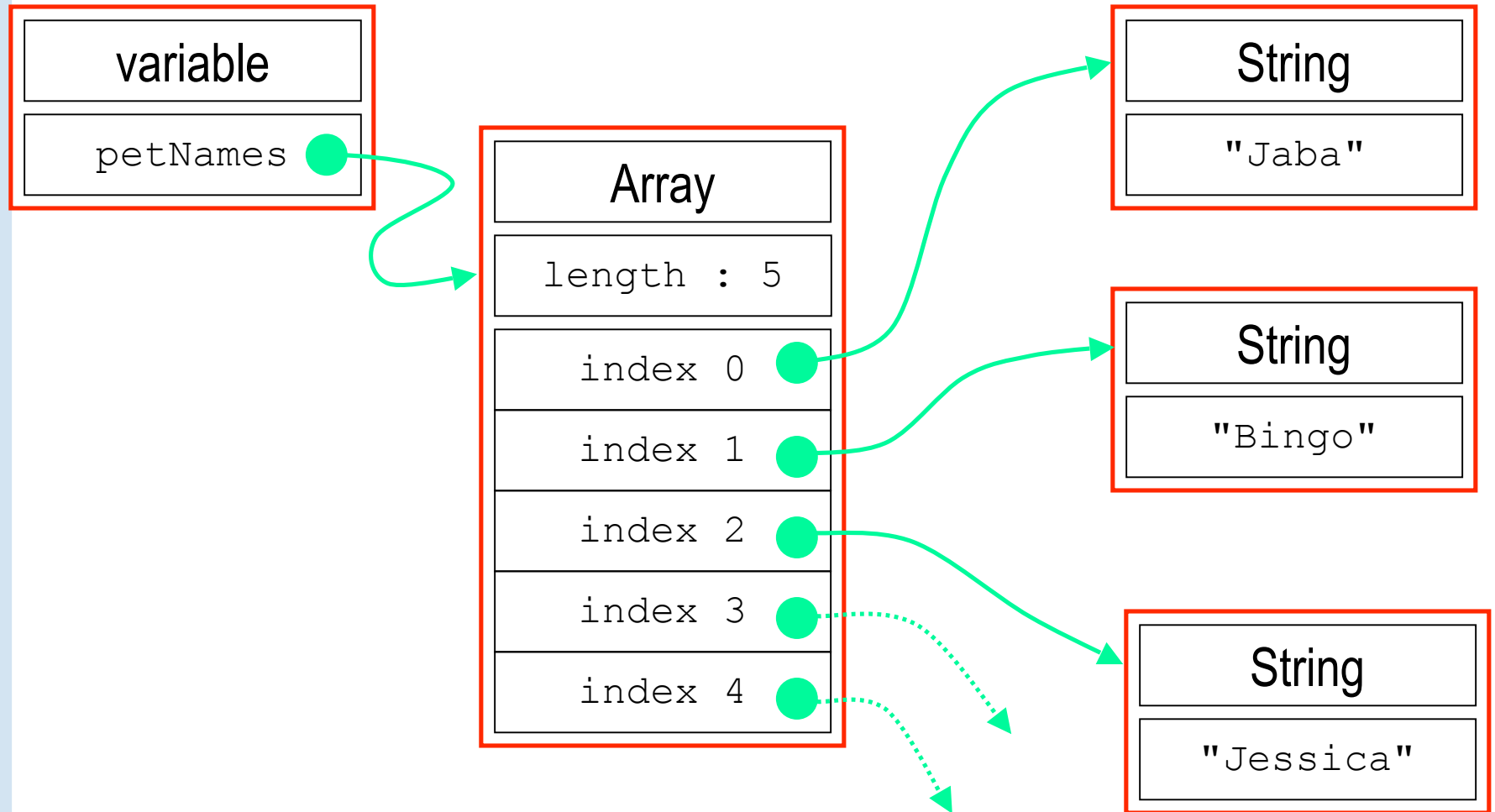


JavaScript Indexed Arrays

- An indexed array is a data type that stores a collection of values, accessible by number
 - » the values in the array are called the *elements* of the array
 - » the elements (or values) are accessed by *index*
 - the index of the first value is 0
 - » the values in the array can be any type
 - usually all the values are the same type
 - but they can be different from one another if necessary



Array Example





Array Declaration and Creation

- Arrays can be created several different ways
 - » **var petNames = new Array();**
 - 0-length array with no elements in it yet
 - » **var studentNames = new Array(102);**
 - 102-element array, all of which have the value *undefined*
 - » **var myList = ["Sally", "Splat", "Google"];**
 - 3-element array initialized with an *array literal*
- Arrays have a property that stores the length
 - `<array name>.length`
 - » you can lengthen or shorten an array by setting the length to a new value



Array Element Access

- Access an array element using the array name and position:
<array name> [<position>]
- Details:
 - » *<position>* is an integer expression.
 - » Positions count from zero
- Update an array element by assigning to it:
<array name> [<position>] = <new element value> ;

```
myCurrentCarNo = carList[carList.length-1];  
myCurrentCar = carList[myCurrentCarNo];
```

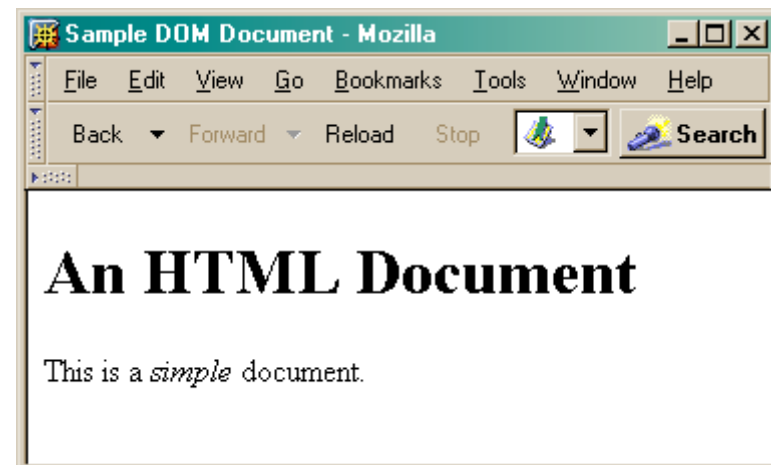

What the heck is the DOM?

- Document Object Model
 - » Your web browser builds a *model* of the web page (the *document*) that includes all the *objects* in the page (tags, text, etc)
 - » All of the properties, methods, and events available to the web developer for manipulating and creating web pages are organized into objects
 - » Those objects are accessible via scripting languages in modern web browsers

This is what the browser reads (sampleDOM.html).

```
<html>
  <head>
    <title>Sample DOM Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.
  </body>
</html>
```

This is what the browser displays on screen.



This is a drawing of the model that the browser is working with for the page.

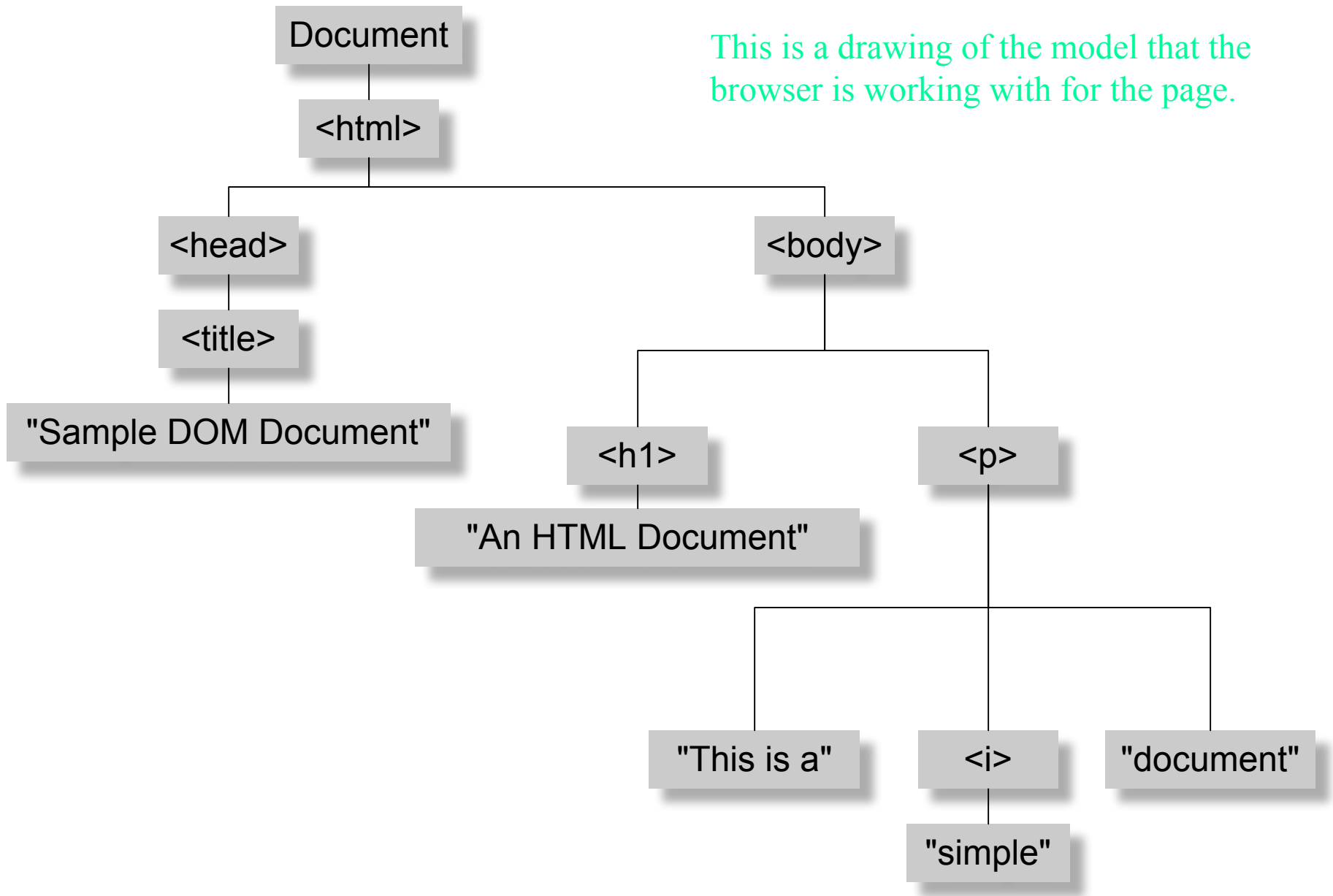


Figure 17-1. The tree representation of an HTML document
Copied from JavaScript by Flanagan.



```
document.getElementById("radioLC").checked
```

- **document**
 - » The root of the tree is an object of type `HTMLDocument`
 - » Using the global variable `document`, we can access all the nodes in the tree, as well as useful functions and other global information
 - `title`, `referrer`, `domain`, `URL`, `body`, `images`, `links`, `forms`, ...
 - `open`, `write`, `close`, `getElementById`, ...
- **getElementById("radioLC")**
 - » This is a predefined function that makes use of the `id` that can be defined for any element in the page
 - » An `id` must be unique in the page, so only one element is ever returned by this function
 - » The argument to `getElementById` specifies which element is being requested
- **checked**
 - » This is a particular property of the node we are looking at, in this case, a radio button
 - » Each type of node has its own set of properties
 - for radio button: `checked`, `name`, ...
 - refer to the HTML DOM for specifics for each element type
 - Some properties can be both read and set



How to organize the data?

- Before relational databases (the kind we study) there were only “flat files”
 - » Structural information is difficult to express
 - » All processing of information is “special cased”
 - custom programs are needed
 - » Information repeated; difficult to combine
 - » Changes in format of one file means all programs that ever process that file must be changed
 - eg, adding ZIP codes



Relational Databases

- Information is stored in tables
 - » Tables store information about *entities*
 - » Entities have characteristics called *attributes*
 - » Each row in a table represents a single entity
 - Each row is a set of attribute values
 - Every row must be unique, identified by a *key*
 - » Relationships -- associations among the data values are stored

Table structure = *schema*
Table contents = *instance*

A Table in a Database

Tables have names, attributes {fields}, entities {rows}

ID	Last	First	JobID	Hire	Street	City	State	Country
1	Davalino	Nancy	0	5/1/1992	507 20th Ave E	Seattle	WA	USA
2	Fuller	Andrew	3	8/14/1992	908 W. Capital Way	Seattle	WA	USA
3	Wooster	Berton	1	4/1/1993	722 Moss Bay Blvd	Seattle	WA	USA
4	Peacock	Margaret	2	5/3/1993	4110 Old Redmond Rd	Kirkland	WA	USA
5	Buchanan	Steven	3	10/17/1994	13 Garrett Hill	Seattle	WA	USA
6	Sullimani	Okan	2	12/12/1994	Coventry House	Seattle	WA	USA

Schema for Example table:

ID	number	unique number(Key)
Last	text	person's last name
First	text	person's first name
JobCode	number	current position
Hire	date	first day on job
...		

instance

schema

Redundancy in a database is Very Bad

- Not every assembly of tables is a good database
- Repeating data is a bad idea
 - » Replicated data can differ in its different location
 - Inconsistent data is worse than no data
 - Cut down on the typos and mis-keyed entries
 - » Keep a *single copy* of any data
 - Reduces memory and data processing costs
 - if it is needed in multiple places, associate it with a key and store key rather than the data
 - » Effort to update is high





Relational Algebra: Tables From Tables

- There are five basic “algebraic” **operations** on tables:
 - **Select** -- pick rows from a table
 - **Project** -- pick columns from a table
 - **Union** -- combine two tables w/like columns
 - **Difference** -- remove one table from another
 - **Product** -- create “all pairs” from two tables

From this basis, many more complicated operations can be built up



Database Structure

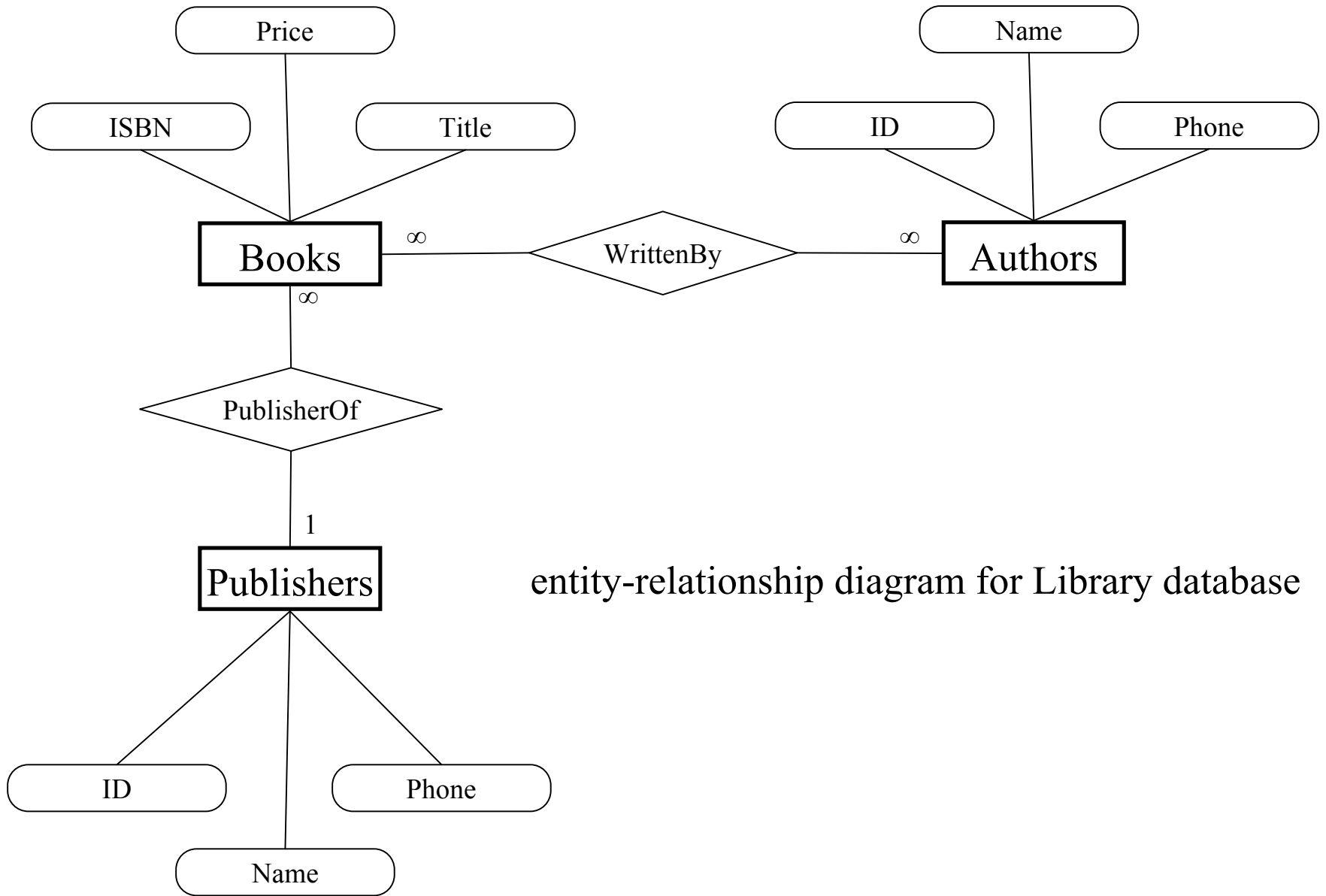
- StructuredQueryLanguage (SQL) is the language we talk to the database in
- A database contains one or more *tables*
 - » Tables include *entities* with *attributes*
 - » There are *relationships* defined between the entities in the various tables
 - » Retrieve information from the tables using *queries*
 - » Create GUI front ends (*forms* and *reports*) for users
- First, design the database or create the schema
 - » What are the entities?
 - » What are the attributes of each entity?
 - » What are the relationships between tables?

SQL behind the scenes

```
All Books w/ all fields : Select Query
SELECT books.*, publishers.*
FROM publishers INNER JOIN books ON publishers.ID=books.PubID;
```

```
Title & Publisher : Select Query
SELECT books.ISBN, books.Title, publishers.Name
FROM publishers INNER JOIN books ON publishers.ID=books.PubID;
```

```
Costly books : Select Query
SELECT books.ISBN, books.Title, books.Price, publishers.Name
FROM publishers INNER JOIN books ON publishers.ID = books.PubID
WHERE (((books.Price)>15));
```



entity-relationship diagram for Library database



Front end and Back end

- Front end
 - » We present the data to the user with some sort of Graphical User Interface
 - Simple tabular display as we have been doing
 - MS Access provides *Forms* and *Reports* for GUIs
 - Web pages
- Back end
 - » The database stores the data in tables
 - » We use queries to construct new "virtual" tables

Forms & Reports are just a Face for a Table

- The form/report lets the designer arrange the data, label it, provide some control over events, etc
 - » the **presentation**
 - » multiple presentations are possible depending on the specific needs of each user
- Underlying data comes from a table or a query
 - » the **content**
 - » single source of data ensures consistency



Tables are not pretty ...



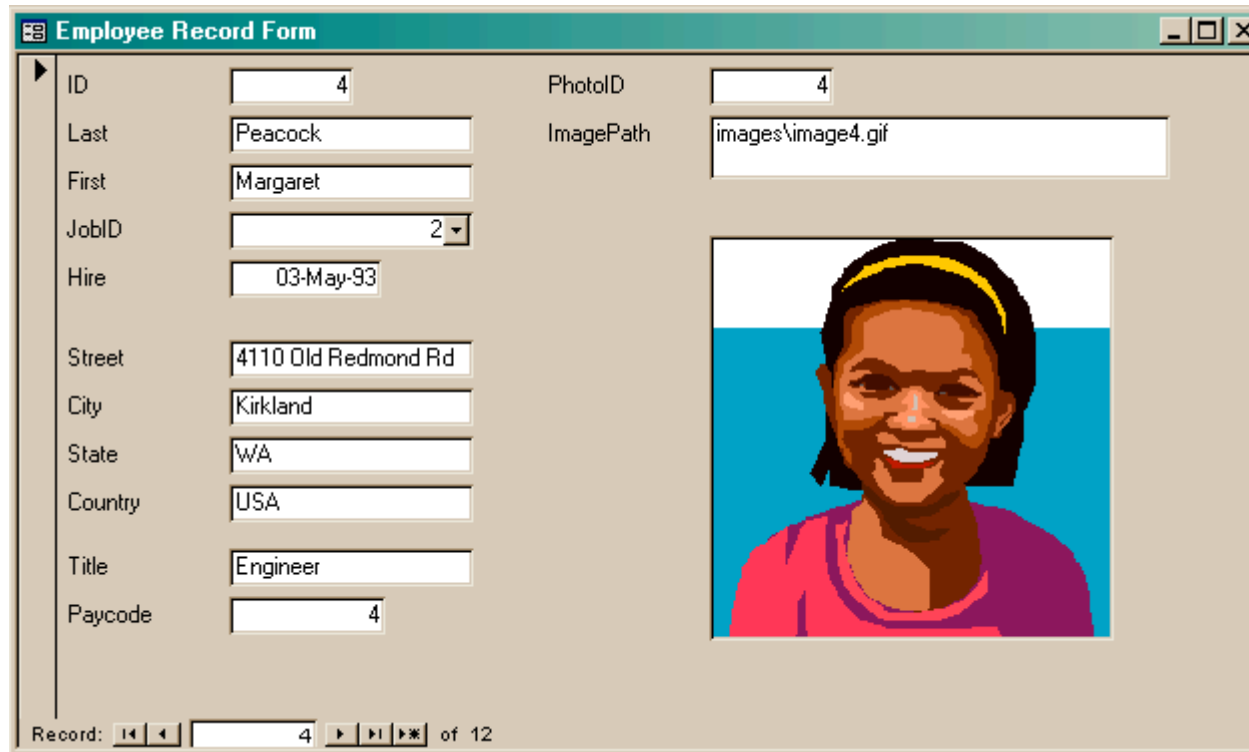
Users need help understanding what they are looking at and what they can do with it ...

... so we developed Forms for controlling the display of data for the user who is reviewing or updating specific records.



Views as Forms

A form is primarily used to enter or display data in a database



The screenshot shows a window titled "Employee Record Form" with a light beige background. The form contains the following fields and values:

ID	4	PhotoID	4
Last	Peacock	ImagePath	images\image4.gif
First	Margaret		
JobID	2		
Hire	03-May-93		
Street	4110 Old Redmond Rd		
City	Kirkland		
State	WA		
Country	USA		
Title	Engineer		
Paycode	4		

At the bottom of the form, there is a record navigation bar showing "Record: 4 of 12" with navigation icons for first, previous, next, and last records.

On the right side of the form, there is a photo of a woman with dark hair, wearing a yellow headband and a pink top, smiling against a blue background.

Last lecture we developed Forms for better display to the user while updating the table.

Forms are not very compact ...



One Portals Way, Twin Points WA 98156
 Phone: 1-206-555-1417 Fax: 1-206-555-5938

Ship To: Rattlesnake Canyon Grocery
 2817 Milton Dr.
 Albuquerque NM 87110
 USA

Bill To: Rattlesnake Canyon Gr
 2817 Milton Dr.
 Albuquerque NM 87110
 USA

Users like to have reports densely packed with information and logically arranged ...

Order ID:	Customer ID:	Salesperson:	Order Date:	Required Date:	Shipped Date:	Ship Via:
11077	RATTC	Nancy Davolio	06-May-1998	03-Jun-1998		United Package

Product ID:	Product Name:	Quantity:	Unit Price:	Discount:	Extended Price:
2	Chang	24	\$19.00	20%	\$364.80
3	Aniseed Syrup	4	\$10.00	0%	\$40.00
4	Chef Anton's Cajun Seasoning	1	\$22.00	0%	\$22.00
6	Grandma's Boysenberry Spread	1	\$25.00	2%	\$24.50
7	Uncle Bob's Organic Dried Pears	1	\$30.00	5%	\$28.50