* Web App
  + Front End BackEnd Database
* What they see does some things stores info
* Fb
* Ruby on Rails-MVC FRAMEWORK
* 6 TIPS in Learning
  + believe in yourself (5-7 days, fully working web site)
  + Stick to our course!
  + Don’t skip assignments (learning comes by doing them\_
  + Listen to your mento (code reviews, feedbacks, etc)
  + Get Certified (jobs etc)
* FAQ
  + Can I really learn all these? Yes
  + Start from first course CS101 and start from there (1, 2 hr)
  + Do the assignments
* Full stack programmer: front-end, back-end.
* <file:///Users/borislavparov/Dev/CodingDojo13/html/assignment_3/assignment3.html>
* Web application: is just bunch of forms users give us info, we store it in the database and maybe later we display it to the users.
* -from: web appl is collection of PHP function and handling form info. Ruby and Python have forms and works exact same way.
* POST- is a one way use. It passes info from index to process page.
* SESSION- session\_start(); on beginning of our page
* Has to be an association, associative array
* Every time I create a session it has to be with name. String name.
* require\_once(‘conn.php’);
* While ($row = mysql\_fetch\_assoc (results)) call and go through each loop.
* Web Page
* Dynamic-is content on my page that I want to change.
* Static-is content that never changes.
* 1. Form page-where is it going?
* 2. Process page
  + 1. Validate $\_POST
  + 2. (Insert)
  + 3 (Set $\_SESSION)
  + 4. Redirect: header(“Locaton: index.php”);
* $\_SESSION- used to store temp info relative to session. It is used at end of
* can be used if it is referenced in another page.
* I can also unset errors.
* Set $\_SESSIONS variables only to
* HTML/CSS...
* 1. create first the blueprint
* - plot the wrapper
* - plot the major sections (divs) and put a colored border using css
* - have those divs in their proper place by using floats and by adjusting widths and heights
* 2. then you can go to the details, one section at a time
* - u may start with putting background color first per section (should u need to add one)
* - then plot the contents in html
* - put the styles using css
* 3. then move to the next section
* -u may start with the header (top part of the page)
* -then the left part
* -then right
* -then footer
* HTML is how we display info to the web site
* CSS is makes the presentation prettier!
* CSS (Cascading Style Sheets)
* Headings:
* <h1> Title page
* <h1 to h6 are headings
* Use lower case in HTML
* No camel case (combination of lower and upper case)
* Alignment and indentation (use to easily identify the different rows)
* Id=”wrapper”> (it indents all text)
* CSS-makes the web page interactive and pretty
* ->can get lost very easily
* ->CSS tags to know
* ->google CSS tutorial
* **Important tags you will have to know**
  + Introduction (html)
  + Head (title, meta description tag, meta keyword tag)
  + Body:
* 1. Table (table, thead, tbody, tr, th, td)
* 2. Images, links, and paragraphs (img, a, p)
* 3. Lists (ul, li, ol)
* 4. Others (b, i, u)
* 5. Headings (h1, h2, h3)
* 6. Sections (div, span; difference between **id** and **class**)
* 7. Form (form, input text, input checkbox, input radio, input hidden, input submit, selection/option, button)
* Never do in line styling (makes coding looks messy)
* Do it internal: <link.rel=”stylesheet”type=”text/css” href=/css/main css”/>
* Can drag files from my drive unto “Sublime text” to work on it.
* Localhost/coding\_dojo/css/index.html (to view the sublime text code I did)
* Can’t drag the file to the browser
* MAMP-is my server on my own computer (apache takes it and sends it to the browser)
* [www.zengarden.com](http://www.zengarden.com)
* same HTML
* Ginza Network-start up
* White combinatory
* …/ go in directory above where I am.
* [www.w3schools.com](http://www.w3schools.com) good tutorials
* <http://w3schools.com/css/css_boxmodel.asp>
* Marging
* Border
* Padding
* Content
* I need to make sure that I link my CSS file put it in the html folder, specify location by linking it.
* CSS Display and Visibility
* <http://w3schools.com/css/css_display_visibility.asp>
* by display we can hide contents
* to position HTML elements use flots
* h1, h2, h3, h4, h5 shows the size of the header in HTML
* id-only once, appears
* class-appears more than once
* margin: give space out side the element
* padding: give space in side the element
* **Fundamentals**
  + CSS syntax (selectors, declaration (property: value))
  + CSS id and class
  + Internal style sheet vs External style sheet vs inline (don't do!)
  + background (background-color, background-image, background-repeat)
  + text (font-size, font-weight, color, font-family, font-style, line-height, text-align)
  + Links (a, a:target, hover)
  + Lists (list-style, list-style-type)
  + Box (border, margin, padding, width, height, min-height, min-width) - also explain auto (e.g. margin: 5px auto)
  + Display (display: none/inline/block)
  + Positioning (position: absolute/relative, top, left, cursor, z-index)
  + Float and clear
* #nav ul li a{}
* #main\_content{}
* By using # you can reference the items below.

**Topics-**how to structure a web site

* + CSS selector overview
  + Lego Concept
  + Box Model (width, height, padding, borders, margin)
  + Display (inline, inline-block, block)
  + Float (left, right)
  + 🡪 Clear

Document

<div>

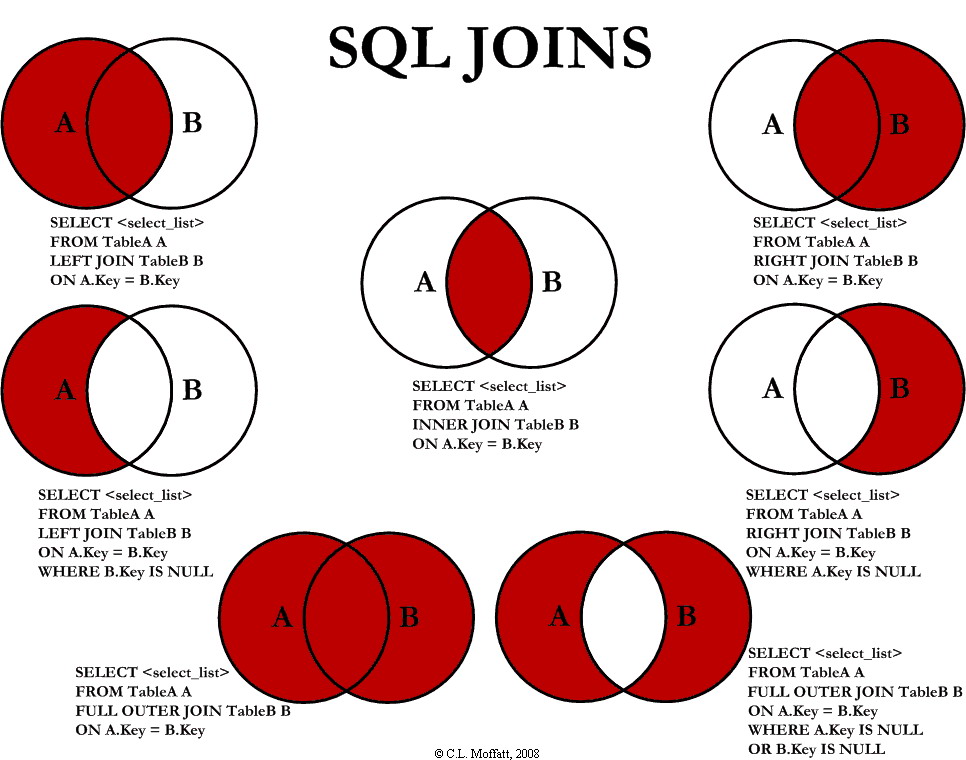
<p> . . . </p>

<a> . . . </a>

* Mark ups-describe where info need to go
* Hypertext-mark up language.
* Many ways to mark data
* Tag:
* <html>. . . </html>
* <head>
* <title>. .. </title>
* you put styling

you put scripts

meta (above) text (specify author, when it was devlpd)

* Java Script comes after CSS, so it’d define where the blox is.
* </head>
* <ul>
* <li>means unordered list </li>
* <li> leas items
* </ul>
* <ol> means ordered list </ol>
* <img src=”images/beach.png”>
* src means source
* <a> means ‘anker’ very critical
* href-hypertext reference
* <a href=”ecommerce html”>ecommerce </a>
* \*\*\***At beginning of every CSS file type that on top since margin and padding is set by default**\*\*\*
* \*{
* padding: 0px;
* margin: 0px;
* font-size: 100%;
* font-family: Arial;
* vertical-align: baseline;
* }
* /\*the line height depends on what you need, it can be 1.2, 1.4, until 2 or 3\*/
* body{
* line-height: 1.5;
* }
* **jquery functions:** .addclass, .after, .append, .attr, .before, .html, .text, .val, .toggle, .hide, .show, .slideDown, .slideToggle, .slideUp, .fadeOut, .fadeIn, .submit, .serialize, .focus, .click. After you build this program, upload your codes below.
* Visit <http://docs.jquery.com/Main_Page>
* search and study the following codes.
* **Effects** (functions to do some cool animation effects)
* .hide()
* .show()
* .toggle()
* .slideDown()
* .slideToggle()
* .slideUp()
* .fadeOut()
* .fadeIn()
* **Forms** (functions related to information in the form)
* .focus()
* .change()
* .serialize()
* .submit()
* **CSS** (adding or removing a class for any html element/DOM)
* .addClass()
* .removeClass()
* **Manipulation** (retrieving or setting value or text in any html element)
* .after()
* .append()
* .attr()
* .before()
* .html()
* .text()
* .val()
* **Traversing** (ways of accessing another html element nearby the one that the event is triggered)
* .children()
* .each()
* .parent()
* .siblings()
* **Events** (functions to handle an event)
* .click()
* .live()
* .submit()
* Again, the best way to learn is to study these tags, watch some of our videos, and to start working on the assignments. Your mentor can help you if you are stuck.
* 1. $(this).serialize()
* 2. DOM-> presents, children, siblings
* 3. var, if else
* <div>
* <img id=”cat” sile””>
* <img id=”dog” sil””>
* </div>
* .submit is only for forms.
* -> Jquery:
* $(this).serialize();
* you tell the <div> to target only that peticular <img> by using $(this)
* $(‘form’).submit(function(){
* alert ($($(this)’p’)
* <input type=”text” name=”first\_name”>
* <input type=”text” name=”last\_name”>
* ->validation: first\_name=Boris & last\_name=Parov
* return false; means stop any process until I find what is wrong
* **DOM: document object model-**visual manipulation of how I add or remove things in JS.
* You create “event listeners” for the DOM
* $(‘a’).click(functions(){});
* jQuery-$(this)
* HTML codes
* **<img src="image1.jpg" />**
* **<img src="image2.jpg" />**
* **<img src="image3.jpg" />**
* **<img src="image4.jpg" />**
* **<img src="image5.jpg" />**
* Let's say that you wanted to make your app work such that when the user clicks on an image, that particular image disappears.  How would you do this?
* One way is to change your html so that we add a unique ID for each image and add a jquery code like below.
* **<html>**
* **<head>**
* **<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js"></script>**
* **<script>**
* **&nbsp; $(document).ready(function(){**
* **$('#image1').click(function(){**
* **$('#image1').hide();**
* **});**
* **$('#image2').click(function(){**
* **$('#image2').hide();**
* **});**
* **$('#image3').click(function(){**
* **$('#image3').hide();**
* **});**
* **$('#image4').click(function(){**
* **$('#image4').hide();**
* **});**
* **$('#image5').click(function(){**
* **$('#image5').hide();**
* **});**
* **});**
* **</script>**
* **</head>**
* **<body>**
* **<img src="image1.jpg" id="image1" />**
* **<img src="image2.jpg" id="image2" />**
* **<img src="image3.jpg" id="image3" />**
* **<img src="image4.jpg" id="image4" />**
* **<img src="image5.jpg" id="image5" />**
* **</body>**
* 1. Updated codes of the above ones: $(this).hide();
* <html>
* <head>
* <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js"></script>
* <script>
* $(document).ready(function(){
* //attach a click event listener to all the img tags when the document is ready
* $('img').click(function(){
* $(this).hide();
* });
* });
* </script>
* </head>
* <body>
* <img src="image1.jpg" />
* <img src="image2.jpg" />
* <img src="image3.jpg" />
* <img src="image4.jpg" />
* <img src="image5.jpg" />
* </body>
* **MySQL...**
* 1. See the structure of the database first
* - check ERD if there's any
* - look at how each table are linked to each other
* - look at the foreign keys and primary keys
* - it would be best if u would also look at the fields and some data in those fields (for each table)
* 2. think of what you need to extract from the database
* - think of it as if you are doing the grocery
* - each aisle could be a table in your databse
* 3. after knowing what u need to extract, identify which table to do u need to extract the data from
* - just like grocery store aisle. u need to identify what aisle that product would belong to
* 4. identify how u could link those tables to get the data you need
* - this is where the primary key and foreign key are very useful
* 5. structure your query.. :)
* - we always use LEFT JOIN in joining tables
* usual format...
* : SELECT
* : FROM
* : LEFT JOIN .. ON (if u have to join tables)
* : WHERE (if there are filter's needed)
* : GROUP BY/ ORDER BY (depending on the need)
* : the ON part is where you are doing the linking
* : e.g.... users.id = posts.user\_id
* Database R-ships
* Unique Identifiers
* Activity
* SQL Demonstration
* **id** is always (+), integer. It’s always ***faster*** than searching by a string (name)
* **JOIN**-is primary mechanism for gathering data from states\_id, cities\_id, businesses\_id.
* **WHERE** states.id=1; (you always want to use the id# if it’s available, otherwise use a string like ‘California’
* **GROUP BY** (I want to display a unique record for this id)
* **1. making table name plural**- make it plural (e.g. users, leads, sites, clients, chapters, courses, modules)
* **2. use ID as primary key**- name it ID (also make it auto-increment)
* 3. when referencing to a primary key in another table name it [singlular name of the table you're referring to]\_id (E.g. user\_id, lead\_id, site\_id, client\_id, chapter\_id, course\_id, module\_id)
* **4. use created\_at and updated\_at** as timestamp
* When we do things in ORM or use Ruby on Rails, it becomes extremely important that we follow these naming conventions.
* Writing queries
* SELECT
* INSERT
* DELETE
* UPDATE
* CREATE
* LEFT JOIN, INNER JOIN, RIGHT JOIN
* GROUP BY
* (Advanced) INSERT ON DUPLICATE KEY
* MONTH()
* MONTHNAME()
* YEAR()
* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* W3Schools has great tutorials on **SQL syntaxes** - please spend 3-5 minutes on each of the pages below to get familiar with the syntaxes.  Do NOT worry about studying other pages.  First master these fundamentals and then if you encounter something you cannot do with these fundamentals (which will rarely happen), you can study other functions to deepen your knowledge in MySQL.  For now, we want you to master these fundamentals first.
* SQL syntax - <http://w3schools.com/sql/sql_syntax.asp>
* SQL select - <http://w3schools.com/sql/sql_select.asp>
* SQL where - <http://w3schools.com/sql/sql_where.asp>
* SQL in - <http://w3schools.com/sql/sql_in.asp>
* SQL and/or - <http://w3schools.com/sql/sql_and_or.asp>
* SQL order by - <http://w3schools.com/sql/sql_orderby.asp>
* SQL count - <http://w3schools.com/sql/sql_func_count.asp>
* SQL sum - <http://w3schools.com/sql/sql_func_sum.asp>
* SQL group by - <http://w3schools.com/sql/sql_groupby.asp>
* SQL insert - <http://w3schools.com/sql/sql_insert.asp>
* SQL update - <http://w3schools.com/sql/sql_update.asp>
* SQL delete - <http://w3schools.com/sql/sql_delete.asp>
* SQL left join - <http://w3schools.com/sql/sql_join_left.asp>
* SQL dates - <http://www.w3schools.com/sql/sql_dates.asp>
* Insert on Duplicate Key Update -<http://dev.mysql.com/doc/refman/5.0/en/insert-on-duplicate.html> -  this could save you lots of time in the future. Later, you'll probably encounter case where you would like to either insert or update a row depending on whether a particular data already exists or not.  This trick could come in handy when you encounter that situation.
  + **SQL JOINS**
* For the difference between left join, right join, inner join, etc, please see <http://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg>
* 
* $\_SESSION-
* Is a bucket where u put staff there the info is available for all the pages for your web application. U can var\_dum to see if info is stored there, but don’t put secure info there b/c it can be hacked. all the info is available for all pass one info from one page to another. Is a variable. Info is stored in my computer
* session\_start
* how to destroy a session
* <?php
* if(isset($\_SESSION['message']))
* {
* echo "<div class='error'>".$\_SESSION['message']."</div>";
* unset($\_SESSION['message']);
* }
* ?>
* Require/Include/md5
* Md5- used to encrypt a password
* Ctype\_alpha- checks to make sure that no numbers are allowed
* It prevents forms from submitting, but still updates.
* The info doesn’t have to be past with.
* You need SESSION to keep track of your info.
* In Ajax, the index.php page never refreshes, but sends a signal to the process.php page.
* \*whenever I write a query on my php copy & past and run it on MySQL to see if it’s working.
* 1. Set database in MySQL
* 2. Set four files: login.php, process.php, connection.php, index.php
  + in index.php:
* use HTML format
* after body enter <form></form>
* process.php: page
* control
* function registerAction()
* function loginAction()
* function messageAction()
* login.php: page
* body>
* <form action=”process.php” method=”post”>
* <input type=”hidden” name=’action’ value=’logout’>
* <input type=’submit’ value=’Log Out’>
* <?= is same as <?php
* connection.php: page
* To call a query use: mysql\_query()
* \*Class
* Instances-are objects
* Objects
* Methods
* Private/Public/Protected
* Properties
* Inheritance
* Construct

=>Protected-makes it available for the children, but blocks if for the instances (public)

* Game: **HAVE** (can have attributes/properties, **DO**-functions/methods)
* -buildings (x, y, width, height)-> properties or attributes
  + Taverns (has ail, do: imporveHealth())
    - For every instance I can have a unique width, height etc.
  + Dungeons (has: low lighting, do: random encounters())
  + Forge (has: weapons; do: trainBlacksmiths())
* All above inherit those characteristics (x, y, width, height coordinates, but also specific characteristics like a blueprint (has ail etc.)
* -People
  + villages
  + elves
  + blacksmith
* \*Blueprint-you can build things with characteristics yet give it specifics.
* \*Instances & Objects are the same
* when attribute is public
* Instances can have its own dimensions, location etc.
* Overview: spend 15-20 hrs.
* If code is simple you can create objects, classes.
* Organize info by classes & objects.
* The value is where code becomes big. You need to know OOP if you wanted to work in large enterprise as a developer.
* MVC-will learn creating classes and objects as in OOP.
* Enterprise level enterprise building-use to build with OOP.
* PHP5 & OOP.
* Know basic functions. Majority programmers don’t use OOP, the new thing b/c they don’t want to learn new things. PHP old procedural method (objects)
* Java, C##,
* Object oriented PHP.
* OO PHP programming you think of software as being virtual objects (mini program in a larger program). PHP (larger program-word blog press), then another object handling (validation), database objects etc. [it better organizes my code].
* OO=Object Oriented
  + Create reusable code-can use in this & other projects
* Frameworks:
* <http://framework.zend.com/>
* <http://pear.php.net/>
* [www.idea22.com](http://www.idea22.com)
* project, zend is OO
* Advantages:
* Any peace of software need be updated and with OO projects are easy to update. Easy for multi-programmer to work at same time. If objects are designed properly others can work on different objects.
* OO project is more robust at beginning, but later is better.
* 3. Develop large libraries to reuse for other projects.
* Ruby, Java, PHP-doing similar things. Using OO can use same code that’ll save lots of time.
* Code libraries in PHP communities: pear.php & framework.zend.com
* 4. OO principles in PHP are consistent with other OO languages.
* 5. Syntax different are minimal.
* 6. OOP-knowing it raises my value as programmer and earn more value. Financial advantages, knowing OOP will increase my value.
  + Disadvantages:

1. it’s more verbal-can use many libraries for OOP
2. if project is more complex or large can go with OO
3. Speed-it’ll be slower, but it wont affect user much, but slower at run time.

* OOP-is about creating modular code
* **Version Control**
  + Keep track of changes to a file or set of files.
* Benefits:
  + Archive stable builds
  + Catch errors where made
  + Go back go old versions
  + Having multiple people work on the same project
* SVN
* **Working Directory**
* **Staging Area**-keeps temporary changes to what file, so we can add it to our git directory (repository)
* **Git Directory** (repository)
* CR (Central repository)-keeps track of each change working on the code
* Enterprise level programmer knows how to use Version Control