Cheatography

WebApps Cheat Sheet

by shlokad via cheatography.com/28400/cs/8345/

General

Full stack: Browser <=> Webserver <=> Database system MEAN stack (Angular, Node, Express, MongoDB)

HTML

Web browsers display Documents described in HTML. Use JavaScript to dynamically generate and update documents. Include directives with content (annotate with tags). **Tags**: formatting info, meaning of text, additional info to display **Malformed html:** complain or figure out missing tag and continue processing **Document:** hierarchical collection of elements **Tags** can have attributes **Entities:** < &g; & " <p>
 <h1>, <b> <i> <pre> (typically used for code, spaces significant) <img> <a href=“”> <table> <tr> <td>   (group related elements, where the group occupies entire lines, forces a line break before and after) Grouping related elements, group is within a single line <form> <input> <textarea> <select> **HTML and XHTML difference:** can skip some end tags, not all attributes have to have values (<select multiple> elements can overlap

URLs

http://host.company.com:80/a/b/c.html?user=Alice&year=2008#p2 **Scheme (http:):** identifies protocol used to fetch the content. **http:** is the most common scheme; it means use the HTTP protocol **https:** is similar to http: except that it uses SSL encryption **file:** means read a file from the local disk mailto: means open an email program composing a message **Host name (//host.company.com):** name of a machine to connect to. **Server's port number (80):** allows multiple servers to run on the same machine. **Hierarchical portion (/a/b/c.html):** used by server to find content. Web server programmed with routing information. Map hierarchical position to function to be performed and possibly the function's parameters **Query parameters** (?user=Alice&year=2008): provides additional parameters **Fragment (#p2):** Have browser scroll page to fragment (html: p2 is anchor tag) **Type of Links Full URL:** 2009 News **Absolute URL:** <a href="/stock/quote.html"> same as http://www.xyz.com/stock/quote.html **Relative URL** (intra-site links): <a href="2008/March.html"> same as http://www.xyz.com/news/2008/March.html Define an anchor point (a position that can be referenced with # notation): <a name="sec3"> Go to a different place in the same page: <a href="#sec3"> **Users of URLs** Loading a page, loading an image, spreadsheet, embed a page: <iframe src="http://www.google.com"> **URL encoding** must be represented as %xx, where xx is the hexadecimal value of the character: **URI vs URL** A URL is a type of URI. But that doesn’t mean all URIs are URLs. The part that makes a URI a URL is the inclusion of the “access mechanism”, or “network location”, e.g. http:// or ftp://.

By **shlokad**
cheatography.com/shlokad/

Not published yet.
Last updated 7th June, 2016.

WebApps Cheat Sheet

by shlokad via cheatography.com/28400/cs/8345/

AngularJS

Events

Types: Mouse-related: mouse movement, button click, enter/leave element, keyboard-related: down, up, press, Focus-related: focus in, focus out (blur), Input field changed, Form submitted, Timer events. **Event handling:**: ... ; element.onclick = mouseClick; element.addEventListener("click", mouseClick); **Event object**: subclassed MouseEvent, KeyboardEvent. type, timestamp, currentTarget (element that listener was registered on), target (element that dispatched the event). **MouseEvent**: button, pageX, pageY (mouse position relative to the top-left corner of document), screenX, screenY (mouse position in screen coordinates). **KeyboardEvent**: keyCode, charCode (integer unicode, if there is one). Draggable rect; elem.style.left = (elem.offsetLeft + (event.pageX - prevX)) + "px"; **Capture phase**: Start at the outermost element and work down to the innermost nested element. ○ Each element can stop the capture, so that its children never see the even event.stopPropagation(). element.addEventListener(eventType, handler, true); **Bubble phase**: Most on handlers (e.g. onclick) use bubble, not onfocus/blur. Invoke handlers on the innermost nested element that dispatches the event (mostly right thing). **Timeout**: setTimeout(myFunc, 5*1000), setInterval(myfunc, 50), clearInterval(token). **Event processed one-at-a-time**: Does not interleave with other js execution. Handlers run to completion, no multithreading.

Single Page Applications

Browser tosses current JS environment when navigating to a different page: Typing into location bar or forward/back buttons. Selecting a bookmarked URL, page refesh, assignments to window.location. Can change hash fragment and query parameters without reload. **Deep linking**: the URL should capture the web app's context so that directing the browser to the URL will result the app's execution to that context. **Two approaches**: maintain the app's context state in the URL, provide a share button to generate deep linking URL. **ngRoute**: A directive (ngView) to indicate where view components should be inserted in template. A service ($route) that watches window.location for changes and updates the displayed view. A configuration ($routeProvider) that allows the user to specify the mappings of URLs to view components. **<div ng-view> . angular.module('cs142App', ['ngRoute']). cs142App.config(['$routeProvider', function($routeProvider) { $routeProvider.when('/Book/:bookId', { templateUrl: 'book.html', controller: 'BookController'}).** Passing parameters**: $routeParams.bookId.

CSS

Separate style from content body { -> selector margin: 8px; property, value **Selectors:** Tag name: h1, class attribute: .large, Tag and class: p.large, Element id #p20 Pseudo selectors: p: hover, a:link, a:visited **Color:** color and background_color rgb(255, 255, 0) **Box model** Margin -> border -> padding -> element (width, height) Total element width: width + left padding + right padding + left border + right border + left margin + right margin **Distance units:** Absolute: 2px, 1mm, 2cm, 0.2in, 3pt 2m (2 times the element's current font size, 2rem (2 times the root element's current font size) border-bottom-color/style/width **Position:** position: static; (default) - Position in document flow
position: relative; Position relative to default position via top, right, bottom, and left properties
position: fixed; Position to a fixed location on the screen via top, right, bottom, and left properties
position: absolute; Position relative to ancestor absolute element via top, right, bottom, and left properties
background-image, background-repeat, font, font-family, font-size, font-weight, font-style, text-align, vertical-align, cursor (eg. help) **display:** none (element not displayed, and takes no space in layout) display: inline display: block visibility: hidden (element is hidden but space still allocated)|visible font-size inherited, border, background not inherited Most specific rule wins <link rel="stylesheet" type="text/css" href="myStyles.css" />

Template HTML with additional markup used to describe what should be displayed **Directive** Allows developer to extend HTML with own elements and attributes (reusable pieces) **Scope** Context where the model data is stored so that templates and controllers can access **Compiler** Processes the template to generate HTML for the browser **Data Binding** Syncing of the data between the Scope and the HTML (two ways) **Dependency Injection** Fetching and setting up all the functionality needed by a component **Module** A container for all the parts of an application **Service** A way of packaging functionality to make it available to any view **Controller**: angular.module("cs142App", []) .controller('MyCntrl', function($scope) { $scope.yourName = ""; $scope.greeting = "Hola"; }); Each template component gets a new scope and is paired with a controller. A scope object gets its prototype set to its enclosing parent scope. **Dot in your model**: Model reads will go up to fetch properties from inherited scopes. Writes will create the property in the current scope! **Scope digest and watches**: Two-way binding works by watching when expressions in view template change and updating the corresponding part of the DOM. Angular add a watch for every variable or function in template expressions. Compared to previous values during digest processing. **Add your own watches:** ($scope.$watch('firstName', function())) (e.g. caching in controller) **Trigger a digest cycle:** ($scope.$digest()) (e.g. model updates in event) **ngRepeat**: <li ng-repeat="person in peopleArray". **ng-if**: Include in DOM if expression true. (will create scope/controllers when true, exit goint to false) **ng-show**: Like ngIf except uses visibility to hide/show DOM elements. Occupies space in DOM structure (but not on screen) when hidden. Scope & controllers created at startup. **ng-click**: run code when element clicked. **ngModel**: Bind with input, select,

textarea tags. ng-href, ng-src. **ng-include**: . **Directives**: Package together HTML template and Controller and extend templating language. Be inserted by HTML compiler as: attribute (...) or element (<my-dir arg1="foo">...</my-dir>). Specify the template and controller to use. Accept arguments from the template. Run as a child scope or isolated scope **Services**: Used to provide code modules across view components. $http, $resource, $xhrFactory, $location, $window, $document, $timeout, $interval. **Dependency injection**: var cs142App = angular.module('cs142App', ['ngRoute']); cs142App.config(['$routeProvider', function($routeProvider) { cs142App.controller('MainController', ['$scope',function($scope) {. **Angular APIs**: **ngRoute** - Client-side URL routing and URL management, ngResource (REST API access), ngCookies, ngAria (Support for people with disabilities), ngTouch, ngAnimate, ngSanitize (parse and manipulate HTML) safely.

WebApps Cheat Sheet

by shlokad via cheatography.com/28400/cs/8345/

Javacript basics

Front End Programming

CGI: Certain URLs map to executable programs that generate web page. **Templates**: mix code and HTML. Write HTML document containing parts of the page that are always the same. Add bits of code that generate the parts that are computed for each page.**Object-relational mapping**: simplify the use of databases (make database tables and rows appear as classes and objects). **Model View Controller**: Model: manages the application's data, view: HTML/CSS, controller: fetch models and control view, handle user interatcions. **AngularJS view template**: Hello {{models.user.firstName}}. **Angular controller**: Angular creates $scope and calls controller function called when view is instantiated. **Angular data**: REST APIs and JSON. **Mongoos's ODL**: var userSchema = new Schema({ firstName: String, lastName: String, }); var User = mongoose.model('User', userSchema);

Responsive Web Apps

Add grid layout system with relative rathen than absolute measures. Add @media rules based on screen sizes. Made images support relative sizes. @media only screen and (min-width: 768px) { / *tablets and desktop layout* / } @media only screen and (max-width: 767px) {/ *phones* / } @media only screen and (max-width: 767px) and (orientation: portrait) { / *portrait phones* / }. **Use CSS breakpoints to control layout and functionality**: Layout alternatives, App functionality conditional on available screen real estate.

Javascript Programming

OOP: In methods this will be bound to the object. Object.keys(this). In non-method functions, this is the global object or undefined if strict. **Functions** can have properties too. plus1.invocations. **Classes**: var r = new Rectangle(26, 14). **Prototypes**: Prototype objects can have prototype objects forming a prototype chain. On an object property read access JavaScript will search the up the prototype chain until the property is found. The properties of an object are its own property in addition to all the properties up the prototype chain. This is called prototype-based inheritance. Property updates always create property in object. Rectangle.prototype.area = function() {}. Object.keys() would return width and height. **Functional**: can write entire program as functions with no side effects. **Closures**: **For private properties** var myObj = (function() { var privateProp1 = 1; var privateProp2 = "test"; var setPrivate1 = function(val1) { privateProp1 = val1; } var compute = function() {return privateProp1 + privateProp2;} return {compute: compute, setPrivate1: setPrivate1}; })(); typeof myObj; // 'object' Object.keys(myObj); // ['compute', 'setPrivate1'] **Fix for fileNo**: Make fileno an argument. printFileLength(fileNo); **JSON**: JSON.stringify(obj) or JSON.parse(s).
Bad parts: Declares variables on use. Automatic semicolon insertion - Workaround: Enforce semicolons with checkers. Type coercing equals: ==. Always use ===, !==. ("" == "0") is false but (0 == "") is true, so is (0 == '0') (false == '0') is true as is (null == undefined).**Assign default value**: hostname = hostname || "localhost";. **Access an undefined object property**: var prop = obj && obj.propname;. Handle multiple this. var self = this; fs.readFile(self.fileName + fileNo, function (err, data) { console.log(self.fileName,fileNo); });

high-level, dynamic, untyped, and interpreted programming language ... is prototype-based with first-class functions, … ... supporting object-oriented, imperative, and functional programming
Need to define variables, variables have the type of the last thing assigned to it. Undefined, number, string, boolean, function, object. Two scopes: Global and function local. All var statements hoisted to top of scope.
Number: NaN, infinity, bitwise operators 32 bit. **String**: indexOf(), charAt(), match(), search(), replace(), toUpperCase(), toLowerCase(), slice(), substr() **boolean**: false, 0, "", null, undefined, and NaN **undefined and null**: null - a value that represents whatever the user wants it to Use to return special condition (e.g. no value) typeof null == 'object' Both are falsy but not equal (null == undefined; null !== undefined) **functions**: arguments array. All functions return a value (default is undefined). **first class functions**: can be passed as arguments, returned, stored. **Object**: Object is an unordered collection of name-value pairs called properties. Name can be any string. bar.name or bar["name"]. foo.nonExistent == undefined. **Global scope** object in browser (window[prop]). To remove properties, use delete. delete foo.name. **Arrays**: Objects. Can be sparse and polymorphic. push, pop, shift, unshift, sort, reverse, splice. Can store properties like objects. **Dates**: var date = new Date(). number of ms since midnight Jan 1st, 1970 UTC. valueOf(),, toISOString(), toLocaleString(). **Regular Expressions**: var re = /ab+c/; or var re2 = new RegExp("ab+c"). String: search(), match(), replace(), split(). RegExp: exec() and test(). /HALT/.test(str); // Returns true if string str has the substr HALT /halt/i.test(str); // Same but ignore case
/[Hh]alt [A-Z]/.test(str); // Returns true if str either "Halt L" or "halt L" 'XXX abbbbbbc'.search(/ab+c/); // Returns 4 (position of 'a')

```
'XXX ac'.search(/ab+c/); // Returns -1, no match
'XXX ac'.search(/abc/); // Returns 4
'12e34'.search(/[\d]/); // Returns 2
'foo: bar;'.search(/...\s.:\s...\s;/); // Returns 0
var str = "This has 'quoted' words like 'this'";
var re = /'[']/g;
re.exec(str); // Returns ["'quoted'", index: 9, input:
re.exec(str); // Returns ["'this'", index: 29, input: …
re.exec(str); // Returns null
str.match(/'[']/g); // Returns ["'quoted'", "'this'"]
str.replace(/'[']*/g, 'XXX'); // Returns:
'This has XXX words with XXX.'
try, catch, finally
type="text/javascript"
src="code.js"></script>
```

Cheatography

WebApps Cheat Sheet

by shlokad via cheatography.com/28400/cs/8345/

DOM

window, this. Rooted at window.document.head. nodeName property is element type (uppercase P, DIV, etc.) or #text. parentNode, nextSibling, prevSibling, firstChild, lastChild. Can do element.setAttribute. **By id** document.getElementById("div42"), getElementsByClassName(), getElementsByTagName(). **textContent**: text content of a node and its descendants. **innerHTML**: HTML syntax describing the element's descendants. **outerHTML** same but includes element. getAttribute()/setAttribute(). element.innerHTML = "This text is important", replaces content but retains attributes. **element.style.display = "none"**, image.src = "newImage.jpg". element.className = "active"; document.createElement("P"), document.createTextNode. parent.appendChild(element); or parent.insertBefore(element, sibling), node.removeChild(oldNode); **Redirect to new page**: window.location.href = "newPage.html". **DOM Coordinate System**: The screen origin is at the upper left; y increases as you go down. **The position of an element is determined by the upper-left outside corner of its margin.** Read location with element.offsetLeft, element.offsetTop. Coordinates are relative to element.offsetParent, which is not necessarily the same as element.parentNode. **Default offsetParent is the body element. Position explicity**: Absolute the element no longer occupies space in document flow. **The origin inside an offsetParent (for positioning descendants) is just inside the upper-left corner of its border**. **Absolute**: explicitly positioned, **relative** element is positioned automatically the usual way. This element will become the offsetParent for all its descendants unless overridden. **Dimensions**: Reading dimensions: element.offsetWidth and element.offsetHeight Include contents, padding, border, but not margin. Updating dimensions: element.style.width and element.style.height. **If you want child to be positioned wrt to you, you should be relative, child should be absolute**.

Building web applications

Be consistent, provide context, be fast. Style guide: covers the look and feel of the app. style, user interactions, layout. Design templates: Master-detail template. Front-end-frameworks (bootstrap) css style sheets, html components (buttons, menus, toolbars, lists, table, forms), javascript (modals, transitions, dropdowns). <md-toolbar> layout="row" flex>. <md-content> contents can be the ng-view directive.
Angular Material responsive support: <md-button hide-gt-sm... <md-menu show-lg .. <md-sidenav md-is-locked-open="$mdMedia('gt-sm')". **On mobile**: md-is-locked-open="$mdMedia('gt-sm')". Make a button in the toolbar for opening the nav bar <md-button hide-gt-sm ng-click="toggleUserList()" > <md-icon md-svg-icon="menu" > </md-icon></md-button>toggleUserList = function() { $mdSidenav("users").toggle(); }. **ARIA**: Add text descriptions for things that need it <a aria-label="Photo of user {{user.name}}". **Internationalization**: <h1> {{i18n.GettingStarted}}</h1> <h1 translate>Getting Started</h1> <h1>{{"Getting Started" | translate}}</h1>. **Testing**: Unit testing, each test targets a particular component. Requires mock components. e2e testing: Run tests against the real web application. Scripting interface into browser used to drive web application. Example: Fire up app in a browser and programmatically interact with it. WebDriver interface in browsers useful for this.