**Computer Science Fundamentals**

**Data Structure Unit**

**Useful fashion in the case of a two dimensional table data structure, the implementation is easily accomplished by using a java array.**

**From EJS: Eloquent JavaScript**

**Data structure and JS**

**Data sets**

**[7, 11, 24, 9, ”3”]**

**0 1 2 3 4**

**Objects**

**Value of the type object are arbitrary collection of properties and we can add or remove these properties as we please.**

**Are way to create an object is by using an early brace notation.**

**Var day 1= {**

**Squirrel: false**

**Event: [“work”, “touched me”, “pizza”**

**};**

**Web development**

**What exactly is RESTFUL Programming?**

**Continue**

**The browser doesn’t know in advance where to submit the information and it doesn’t know in advance what information to submit.**

**Both forms of information are entirely supplied by the server.**

**So, how does this apply to HTTP and how can’t be implemented in practice? HTTP is oriented around verbs and resources.**

**The two verbs in main stream usage are get and post, which I think everyone will recognize. However the HTTP standard define several other such a PUT and DELETE.**

**Using Angular 1 for Technical Interview**

**In-person Interview @ Jetblue Airways.**

**Position Summary**

**The Developer front-End performs development, testing and documentation of sophisticated web mobile user interface applications.**

**Asking for Mathew Ford 646-577.1919**

**Michael Charles worked & Jetblue Airways (2012)**

**About the production website Jetblue Airways.com**

**Angularjs/1.2.28/ angular.min.js**

**Angularjs/1.2.28/ angular-route.mins.js**

**Angularjs/1.2.28/ angular-resource.min.js**

**JQuery/1.6.2/ jquery.min.js/ 1735**

**JQuery/jquery.us 1.8.16. custom.min/ 1542**

**WEEK 1 ADDRES**

**JETBLUE**

**Team 27-01 QEENS PLAZA NORTH**

**Met with Evan Habersham LONG ISLAND CITY, NY**

**Ryan Hallquest 11101**

**Mathew Ford Closest TRANSTATION**

**Madam Sounder QEEN BORO PLAZA**

**Eldo**

**Rey Parma Average Time by car from:**

**House: 1 Hour 🡺 1Hour & 15mins**

**DAD’S House**

**JAMES**

**Current Project talked about**

**Blue eye**

**Jetblue.com**

**Web Development & Design Angular 1 Interview questions**

**What is the different between NGSHOW, NGHIDE & NGIF?**

**NGIF**

**An Angular directive that remove or recreate a portion of the DOM tree based on an expression assigned to NGIF evaluate to a false value then a clone of the element is reinserted into the DOM.**

**Create New scope: True**

**Priority level executed @ 600 level**

**Can be used as a multi-element True**

**NGSHOW**

**An angular directive show or hide the given HTML element based on the expression provided to the NGSHOW attribute, The element is shown or hidden by removing or adding the .ng-hide CSS class is predefined in Angular JS and set the display style to none using the important flag.**

**<div ng-show=”my value”></div>**

**<!.....when scope. My value is truthy(element is visible)…>**

**<!.....when scope. My value is falsy (element is hidden)….>**

**<div ng-show= “my value” class=”ng-hide”></div>**

**Web Development & Design Angular 2 interview question**

**What is the difference between NGSHOW, NG HIDE NGIF directive?**

**Continue……**

**NGSHOW/NGHIDE**

**Will always insert the DOM element, but will display/hide based on the condition.**

**NGIF**

**Will not insert the DOM element until the condition is not fulfilled.**

**NGIF is better when we needed the DOM to be loaded conditionally, as it will help load page a bit faster compared to NGSHOW/NGHIDE.**

**So we only need keep in mind what the difference these directive is, so deciding which one to use totally depends on the task requirements.**

**Web Development and Design**

**AngularJS 2**

**Architectural Overview:**

* **Component, Bootstrap, and the DOM**
* **Directive and pipes**
* **Data Binding**
* **Dependency Injection**
* **Services and other business logic**
* **Data Persistence**
* **Routing**

**Skills CSS animate**

**By Blue, by Red, by White, by yellow, by green, By Navy, by Grey, by Black.**

**Font – awesome**

**Icon- laptop icon- camera**

**Background color**

**By-blue #129adq**

**Kayode. Me**

**Portfolio maintenance and enhancements.**

**/= index.html**

**/pages/= will have detailed info of each project with the possibility of adding a live demo.**

**Web Development and Design JavaScript Fundamental**

**JavaScript Closures**

**JS variables can belong to the local or global private variables can be made possible with closures.**

A function can access all variable defined inside the function, or outside the function.

**Global Variables**

**Function my function ()**

**Var a=4**

**Return a\*a;**

In both example A is a global variable belong to the window object in WEBPAGE. Global variables belong to the window object. Global variables can be used a changed by all scripts in the page.

**Var a=4;**

**Function my function ()**

**Return a\*a;**

**GLOBAL VARIABLES 9999/????HP VARIABLE 1/1 HP**

**Global variables live as long your application (your window/ your web page lives).**

**Local variables live when the function is worked, and deleted when function is finished.**

**From WS3 Schools**

**Suppose you want to use a variable for counting, something and want this counter to be available to all functions, you could try.**

**Web Development and Design**

**JAVASCRIPT CLOSURE**

**Var counter= 0;**

**Function add () {**

**Counter= 1;**

**}**

**Add ();**

**Add ();**

**Add ();**

**Add ()**

**//counter is now equal to 3//**

**The counter can be changed only by the add function, problem the counter is declared the globally and can be changed w/o calling add ().**

**If I declared the counter inside the function, nobody will be able to change it without calling add ();**

**Function add () {**

**Var counter= 0;**

**Counter+= 1;**

**}**

**Add (); //1**

**Add (); //2**

**Add (); //3**

**//The counter should be three but it does not work//**

**Use a nested Function**

**ALL FUNCTION HAVE ACCESS TO THE GLOBAL SCOPE**

**In fact, in JavaScript, all function plus has access to the scope “above” them.**

**In this example, the innermost function plus () has access to the counter variables in the parent function.**

**Function add () {**

**Var counter= 0;**

**Function plus () {counter +=1;}**

**Plus ();**

**Return counter;**

**}**

**This could have the counter dilemma, if we could reach the plus () function from the outside.**

**WEB DECELOPMENT AND DESIGN**

**JAVASCRIPT FUNCTION CLOSURES**

**To get our function access to the counter we should create an IFFY or self-invoking functions.**

**Recap a closure is a function having access to the parent scope even after the parent function has closed.**

**Var add (function () {**

**Var counter= 0;**

**Return function () {return**

**Counter+= 1;}**

**Add ();**

**Add ();**

**Add ();**

**The variable add is assigned the return value of a self-invoking function. The self-invoking function only runs ONCE. It sets the counter to zero (0), and returns a function expression. This way add becomes a function. The “wonderful” part is that it can access the counter in the parent scope. This is called JavaScript closure. It makes it possible for a function to have “private” variables // the counter is now 3.**

**The counter is protected by the scope of the anonymous fiction and can only be changed using the add functions.**

**Understanding fictions**

**JavaScript functions are block of code that may be defined once, and executed several times by calling back the function. These functions are very similar to a sub-routine.**

**Procedure**

**Function keyword**

**Function (x) { parameter**

**Return x\*x**

**}**

**Function (x) { square**

**Return x\*x } square(q)//18**

**This is a function that takes its parameter and times it by two.**

**Note: here we assign it to a variable and pass it a number.**

**Web Development and Design**

**Var square= function (x) {**

**Return x\*x**

**}**

**JAVASCRIPT FUNCTION**

**JAVASCRIPT FUNCTIONS ARE PARAMETERIZED: a function definition may include a list of identifiers, known as parameters that work as a local variable for the body of that function.**

**Function Invocation**

**Provide values or argument for the functions parameters.**

**Functions often use their argument values to compute a return value that becomes the value of the functions. Invocation expression.**

**In addition, to the arguments, each invocation has another value, the invocation context- that is the value of this keyword.**

**Key word**

**Parameter**

**Var.s= function (x) {**

**Return X\*X // Return the square of the variable**

**} return keyword**

**Examples of the key words**

**In the JavaScript the things called this, is the object that “owns” the JavaScript code.**

**Web Development and Design**

**Getting more object oriented**

**Type Script give us the ability to be more object oriented with our data, so let’s create a model**

**Export class Car Part {**

**Id: number;**

**Name: string;**

**Description: string**

**In Stock: number**

**Price: number**

**}**

**Note: were declaring what type each of our properties are. This is Type Script.**

**This will allow our compiler to check our code and ensure we’re getting and setting things properly.**

**We can use our new model like so.**

**Import {component} from @ angular/core;**

**}**

**Export class Car Parts Components {**

**Car parts= [{**

**“id”: 1**

**“name”: “super tires”**

**“description”: these tires are the very best”**

**“In stock”: 5**

**“price”: 4.99**

**WEB DEVELOPMENT AND DESIGN**

**Angular 2 Fundamentals**

**The ways Data can Flow**

**When using a web frame work like Angular, that abstracts your code from HTML, there are a few different ways that data can flow.**

,,, .

,, .

…. .

…… .

.

.

…., /

/

.

.

/

**JavaScript to HTML**

**HTML to JavaScript**

**Both ways**

**JavaScript to HTML**

**<Li Class= “card” \*ngfor= “let car part of the car part”>**

**<div class= “panel-body”>**

**<table class= “product info”>**

**<td>**

**<tr>**

**<h2> {{car part.name |uppercase}}</h2>**

**<p class= “description”> {{car part.description}}</p>**

**<p class= “inventory” \*ngif= “car part.instock>{{ }} </P>**

**<P class= “inventory” \*ngif= “ car part.instock> out of stock**

**</td**

**Web Development & Design Angular 2 Fundamentals**

**Structural Directives**

**Bonus: Simplify a sum**

**Just for fun lets go through ways we could simplify this code.**

**Total car parts**

**Web Development and Design**

**Angular 2 Fundamentals refactoring**

**Splitting out our components**

**Main.ts 🡺NOT SCALABLE**

**Main.ts 🡺we import our first component and bootstrap**

**App.component.ts🡺 this component contain our large header.**

**Car-parts.component.ts🡺 this contains list of car parts.**

**Export statement**

**Is used to export functions, object or primitives from a given file.**

**Syntax**

**Export {name 1, name 2, …………name N}**

**Export default expression;**