README.md

Interview Questions for Front-end Developers

How do you validate your markup and why?

W3C validator. 100% valid code is not always the goal, but it helps to write maintainable code. To ensure the code is rendered consistently cross-browser.

Why use a Doctype?

A proper Doctype triggers standards mode in your browser.

What is CDATA?

Used to be required to escape JavaScript in Doctypes prior to HTML5 in order to validate code.

How do you waste time in JavaScript?

Redraw parts of the DOM, e.g. increase font-size in a loop. See http://www.bennadel.com/blog/2370-Overcoming-Asynchronous-Anxiety-By-Testing-JavaScript-s-Event-Loop.htm

Explain hoisting in JavaScript

In JavaScript function declarations (function foo() {}) and variable declarations (var bar) are 'hoisted' i.e. are silently moved to the very top of the scope.

Within its current scope, regardless of where a variable is declared, it will be, behind the scenes, hoisted to the top. However, only the declaration will be hoisted. If the variable is also initialized, the current value, at the top of the scope, will initially be set to undefined.

```
 var myvar = 'my value';
(function() {
  alert(myvar); // undefined
  var myvar = 'local value'; // only the declararion of the variable (var myvar) was hoisted to
})();
```

Above, the declaration was hoisted, but the initialization (= 'local value') was not, resulting in an undefined error.

What is the difference between var x = 1 and x = 1?

Novice JS programmers might have a basic answer about locals vs globals

Intermediate JS guys should definitely have that answer, and should probably mention function-level scope

"advanced" JS programmer should be prepared to talk about locals, implied globals (defined without var, assumed belongs to window DOM, problem if third-party script uses same var name and uses var, they can conflict), the window object, function-scope, declaration hoisting (if statements do not create new scope, only functions do), and . Furthermore, I'd love to hear about [[DontDelete]], hoisting precedence (parameters vs var vs function), and undefined.

See http://sharkysoft.com/tutorials/jsa/content/031.html http://schalk-neethling.com/2011/07/quick-tip-the-problem-with-implied-globals-in-javascript/ http://www.adequatelygood.com/2010/2/JavaScript-Scoping-and-Hoisting http://stackoverflow.com/questions/1484143/scope-chain-in-javascript

Are objects passed by reference or by value?

By reference. In JavaScript, all objects are passed by reference. When you make a change to a reference to an object, you change the actual object. Primitive types are passed by value.

Is an Array an Object in JavaScript?

Yes, Arrays inherit from Objects, and they have array-specific methods such as sort and length.

Explain bit shifting in JavaScript

Operands are treated as 32-bit integers. http://jsfiddle.net/andyshora/M4xAB/

```
var temp;
temp = 5 >> 1; // 101 -> 10 = 2
temp = 8 >> 1; // 1000 -> 100 = 4
temp = 10 >> 1; // 1010 -> 101 = 5
// test
document.write(temp);
```

What is the difference between using dot notation and bracket notation when accessing an object's property?

```
val = ford.speed; // speed has to be a defined property

val = ford['speed'];
val = ford[ attr ]; // attr is a variable which may for example hold a string for the property
val = ford[ nextAttr() ]; // method call will return before accessing ford object
```

Why might a programmer have defined a variable with a capital letter?

As a constructor for an Object. Instance names should be lower case.

```
// example

var Car = new function(){
    this.wheels = 4;
}
// instance of Car
var ford = new Car();
```

this and its scope

```
drink = 'soda'; //global variable

var Foo = function(){
    this.drink = 'beer'; //property of instance object
    console.log('Foo() -> ' + this.drink);
};

function bar(){
    console.log('bar() -> ' + this.drink);
};

var qux = {
    drink: 'wine', // in the scope of qux, 'drink' = 'wine'
    getDrink: function(){
        console.log('qux() -> ' + this.drink);
    }
};

//now see how "this' differs in each case

var baz = new Foo(); // Foo() -> beer
bar(); // bar() -> soda
qux.getDrink(); // qux() -> wine
```

What is a closure?

A closure is formed when you nest functions, inner functions can refer to the variables present in their outer enclosing functions even after their parent functions have already executed.

What language is JavaScript based on?

ECMAScript = core language that JS is based on.

What's the difference between a variable and a property?

var a = 'hello'; // variable You cant access variables with a . from the owner, you just need to know that they're there.

Properties are the building blocks of objects. foo.bar

They only appear interchangeable if the parent objects are the same

Write a function to add arguments together

```
function sum() {
  var i, l, result = 0;
  for (i = 0, l = arguments.length; i < l; i++) {
    result += parseInt(arguments[i]);
```

```
  }
  return result;
}

sum(1,2,3); // 6
```

And they should invoke it on your array like this (context for apply can be whatever, I usually use null in that case):

```
var data = [1,2,3];
sum.apply(null, data); // 6
```

Why use frameworks?

Good coders code, great coders reuse. Thousands of man hours have been poured into these libraries to abstract DOM capabilities away from browser specific implementations. There's no reason to go through all of the different browser DOM headaches yourself just to reinvent the fixes.

Why use prototype to define methods instead of static methods?

You are extending the constructor function when you use prototype, so it will be available to all the object instances created with the new keyword, and the context within that function (the this keyword) will refer to the actual object instance where you call it.

```
// constructor function
function MyClass () {
  var privateVariable; // private member only available within the constructor fn

  this.privilegedMethod = function () { // it can access private members
    //..
  };
}

// A 'static method', it's just like a normal function
// it has no relation with any 'MyClass' object instance
MyClass.staticMethod = function () {};

MyClass.prototype.publicMethod = function () {
  // the 'this' keyword refers to the object instance
  // you can access only 'privileged' and 'public' members
};

var myObj = new MyClass(); // new object instance

myObj.publicMethod();
MyClass.staticMethod();
```

Public, private and privaledged members in JavaScript

```
//Private

// Constructor
function Kid (name) {
    // Private
    var idol = "Paris Hilton";
}
```

You can delete or replace a privileged method, but you cannot alter its contents.

```
// Constructor
function Kid (name) {
    // Private
    var idol = "Paris Hilton";

    // Privileged
    this.getIdol = function () {
        return idol;
    };
}
```

A static member is shared by all instances of the class as well as the class itself (i.e. the Kid object), but it is only stored in one place. This means that its value is not inherited down to the object's instances

```
// Constructor
function Kid (name) {
    // Constructor code
}

// Static property
Kid.town = "South Park";


// Public
// Constructor
function Kid (name) {
    // Public
    this.name = name;
}
Kid.prototype.getName = function () {
    return this.name;
};

// ---------------------- all examples

// Constructor
function Kid (name) {
    // Private
    var idol = "Paris Hilton";

    // Privileged
    this.getIdol = function () {
        return idol;
    };

    // Public
    this.name = name;
}

// Public
Kid.prototype.getName = function () {
    return this.name;
};

// Static property
Kid.town = "South Park";

// ---------------- usage

 // Create a new instance

var cartman = new Kid("Cartman");

// Access private property
cartman.idol; // undefined

// Access privileged method
cartman.getIdol(); // "Paris Hilton"
```

```
// Access public property
cartman.name; // "Cartman"

// Access public method
cartman.getName(); // "Cartman"

// Access static property on an instance
cartman.town; // undefined

// Access static property on the constructor object
Kid.town; // "South Park"
```

See http://robertnyman.com/2008/10/14/javascript-how-to-get-private-privileged-public-and-static-members-properties-and-methods/

What is Progressive Enhancement?

Progressive Enhancement consists of the following core principles:

basic content should be accessible to all browsers basic functionality should be accessible to all browsers sparse, semantic markup contains all content enhanced layout is provided by externally linked CSS enhanced behavior is provided by [[Unobtrusive JavaScript|unobtrusive]], externally linked JavaScript end user browser preferences are respected

Can you describe the difference between progressive enhancement and graceful degradation?

Describe feature detection. When features are not supported, maintain accessibility. Content should always be accessible, but user experience sacrificed for older browsers. Try to keep all functionality, and try to keep custom solutions compartmentalised where necessary, for example, in external files loaded for specific browsers.

If you have 10 different stylesheets for a given design, how would you integrate them into the site?

File concatenation and minification in the build process. Don't use @import as it results in an additional request.

How would you reduce page load time/perceived load time?

Reduce image sizes Use image sprites Image datauris inline for small things like avatars Concatenate assets Host assets on different hostnames Use CDN for assets Load content async and feedback loading state to user Minify scripts and stylesheets Page data Cache files

What is FOUC? How do you avoid FOUC?

http://www.learningjquery.com/2008/10/1-way-to-avoid-the-flash-of-unstyled-content
http://paulirish.com/2009/avoiding-the-fouc-v3/

What's a doctype do, and how many can you name?

When you use a DOCTYPE declarations in your web pages, you are telling the web browser what version of (X)HTML your web page should be displayed in. The doctype gives the browser a list of supported tags and does not include any deprecated or proprietary tags in the list. You can get away with writing invalid or incorrect HTML code because most web browsers are amazingly forgiving.

To define the language of a section of the document, add the lang attribute to the appropriate element, such as a div element:

How do you serve a page with content in multiple languages?

```
<div lang="fr-CA" xml:lang="fr-CA">
 Canadian French content...
 </div>
 <div lang="en-CA" xml:lang="en-CA">
 Canadian English content...
 </div>
```

Probably use a CMS to serve up different content, but same styles.

What are data- attributes good for?

Storing data in the DOM

What's the difference between Java and JavaScript?

Here are some differences between the two languages:

Java is an OOP programming language while Java Script is an OOP scripting language. Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only. Java code needs to be compiled while JavaScript code are all in text.

Java is a statically typed language; JavaScript is dynamic. Java is class-based; JavaScript is prototype-based. Java constructors are special functions that can only be called at object creation; JavaScript "constructors" are just standard functions. Java requires all non-block statements to end with a semicolon; JavaScript inserts semicolons at the ends of certain lines. Java uses block-based scoping; JavaScript uses function-based scoping. Java has an implicit this scope for non-static methods, and implicit class scope; JavaScript has implicit global scope.

Here are some features that I think are particular strengths of JavaScript:

JavaScript supports closures; Java can simulate sort-of "closures" using anonymous classes. (Real closures may be supported in a future version of Java.) All JavaScript functions are variadic; Java functions are only variadic if explicitly marked. JavaScript prototypes can be redefined at runtime, and has immediate effect for all referring objects. Java classes cannot be redefined in a way that affects any existing object instances. JavaScript allows methods in an object to be redefined independently of its prototype (think eigenclasses in Ruby, but on steroids); methods in a Java object are tied to its class, and cannot be redefined at runtime.

How would you optimize a websites assets/resources?

File concatenation File minification CDN Hosted Caching etc.

Why is it better to serve site assets from multiple domains?

Browsers restrict simultaneous downloads per domain, so you can just add different A records to point to the same location if you like, or create different buckets on a blob storage account.

Advantages of LESS http://tympanus.net/codrops/2012/01/27/modular-front-end-development-with-less/

Variables. Define any variable like so: @color1: #df0290;and use it later in your code: Mixins.Define useful functions with or without parameters Nested rules.Pretty self-explanatory, as I'm sure this is something you've been wishing for in CSS since you started using it

What does & stand for in LESS?

An ampersand (&) refers to the parent rule. So &.category would translate to article h2.category once the LESS code had been compiled.

How would you organize a LESS library for a large project?

/project/css/

- reset.css — resets default browser styling
- grid.less — supplies mixins for a grid system, such as the .col(@width) mixin above
- type.less — supplies mixins for font styling as well as @font-face rules
- colorscheme.less — LESS variables for the design's various colors
- interface.less — mixins for interface features like buttons, forms, and dialogs
- layout.less — design-specific layout of the site
- style.less — the main stylesheet, including all of the above and adding in whatever site-specific styles are otherwise necessary

Example LESS template - colorscheme.less

```
@background:        #ffffff;
@textcolor:         #252525;
@textcolor-strong:  #090909;
@textcolor-em:      #666666;
@textcolor-blockquote:  #aaaaaa;

@accent1:       #2d9681;
@accent2:       #f8a34b;

@warning:       #d4230f;
```

CSS3 Advantages with LESS

Long repetitive CSS3 code such as gradients, keyframe animations, can all be simplified with Mixins, taking in for example keyframe durations, percentages, animation and easing types as parameters. For gradients, start and stop colours, and blending positions can be controlled with variables.

How does the content model differ in HTML4/5? http://www.w3.org/TR/html5-diff/#content-model

Content model is what defines how elements may be nested — what is allowed as children (or descendants) of a certain element.

At a high level, HTML4 had two major categories of elements, "inline" (e.g. span, img, text), and "block-level" (e.g. div, hr, table). Some elements did not fit in either category.

Some elements allowed "inline" elements (e.g. p), some allowed "block-level" elements (e.g. body), some allowed both (e.g. div), while other elements did not allow either category but only allowed other specific elements (e.g. dl, table), or did now allow any children at all (e.g. link, img, hr).

Notice the difference between an element itself being in a certain category, and having a content model of a certain category. For instance, the p element is itself a "block-level" element, but has a content model of "inline".

To make it more confusing, HTML4 had different content model rules in its Strict, Transitional and Frameset flavors. For instance, in Strict, the body element allowed only "block-level" elements, but in Transitional, it allowed both "inline" and "block-level".

To make things more confusing still, CSS uses the terms "block-level element" and "inline-level element" for its visual formatting model, which is related to CSS's 'display' property and has nothing to do with HTML's content model rules.

HTML5 does not use the terms "block-level" or "inline" as part of its content model rules, to reduce confusion with CSS. However, it has more categories than HTML4, and an element can be part of none of them, one of them, or several of them.

Function.prototype.bind

Can you explain how inheritance works in JavaScript?

Explain how scope works.

Then show how objects can inherit properties: http://jsfiddle.net/andyshora/nvcZE/

```
(function() {
    var genericObject = {
        bar : "Hello World",
        get_bar : function() {
            return this.bar;
        }
    };
    var customObject = Object.create(genericObject);
    customObject.bar = "Aloha folks!";
    document.write(customObject.get_bar() + '<br />'); //outputs: "Aloha folks"
    delete customObject.bar;
    document.write(customObject.get_bar()); //fallbacks to the prototype's value, outputs: "Hel
})();
```

When would you use document.write()?

When messing around on jsfiddle or debugging code perhaps!

Challenges

1. Write a function to reverse a string http://jsfiddle.net/andyshora/8zby5/

```
// a function to reverse a string
function reverseString(str){
    if (str===undefined) return false;
    if (str.length < 2) return str;

    return str.split('').reverse().join('');
}
document.write(reverseString("hello"));
```

1. Implement a function to detect palindromes. http://jsfiddle.net/andyshora/WN2Bv/

```
// function to detect palindromes
function isPalindrome(str){
    if (typeof str!=='string') return false;
    if (!str.length) return false;

    var len = str.length;
    var arr = str.split(''); // split string into array
    var breakAt = Math.floor(len/2);

    for (var i=0; i<len; i++){
        if (i===breakAt) break;
        if (arr[i]!==arr[(len-1)-i]) return false;
    }
    return true;
}
document.write(isPalindrome('aaaaabaaaaa'));
```

1. Calculate the number of digits for a given number.
2. Implement a function that calculates square roots http://jsfiddle.net/andyshora/W5Gzj/2/

```
// function to manually calculate square root in javascript
// returns false on error or no integer square root
function calcSquareRoot(x){

    if (typeof x !== 'number') return false;
    if (x===1) return x; // test for 1, quickest = doesnt mess with logic below
    if (i<0) return false;

    for(var i=2; i<=(x/2); i++){
        if (i*i === x) return i;
    }

    return false;
}
document.write(calcSquareRoot(225)); // 15
document.write('<br />');
document.write(calcSquareRoot(226)); // false
document.write('<br />');
document.write(calcSquareRoot(-1)); // false
document.write('<br />');
document.write(calcSquareRoot(1)); // 1
document.write('<br />');
document.write(calcSquareRoot('2')); // false
document.write('<br />');
document.write(calcSquareRoot(4)); // 2
```

1. Sort and concat arrays in a optimal way
2. Guess the two missing numbers in a array with n - 2 length containing 1..n unsorted numbers