# Front End Test

1. **Using native JavaScript write a function that checks if a string is composed of all unique characters.**

/\*\*

\* Using native JavaScript write a function that checks if a

\* string is composed of all unique characters

\*/

function uniqueString(str){

// I wouldn't consider the empty string unique...maybe?

if(str.length < 1){

return false;

}

let strSet = new Set();

for(let x = 0; x < str.length; x++){

if(strSet.has(str.charAt(x))){

return false;

}else{

strSet.add(str.charAt(x));

}

}

return true;

}

1. **Using JavaScript write a variable declaration that uses coercion to define a variable as an empty object or an existing object, if that object already exists. Explain how coercion makes this possible.**

I am not sure what this questions is asking in the first part. I’m having trouble figuring out what is meant by “define a variable as an empty object or an existing object.” I know what coercion is and would explain it as a feature that helps developers convert different types of primitives back and forth for things like equality checks or type casting. So if you add the number 5 and the string “three” together you would get a string “5three” as a result. I personally don’t like to use coercion because there are some very weird rules and results produced from using coercion. In a book, *JavaScript: The Good Parts*, I recall an appendix section highlighting the “bad parts” of javascript. One of these parts was coercion and how it affects equality checks and produces funky results that make little sense.

1. **Using JavaScript use and Immediately Invoked Functional Element (IIFE) to create a name space and create a pseudo private/public pattern for the functions it contains.**

let iife = (function(){

//define function that is public until exposed

let add = function(x, y){

if(validateTypes(x, y)){

return 'Invalid Input';

}else {

return x + y;

}

};

//define function that will remain private

let validateTypes = function(x, y){

return(typeof x !== 'number' || typeof y !== 'number')

};

return {

//expose the 'add' function

add: add

}

})();

console.log('2 + 3 = ' + iife.add(2, 3)); // 5

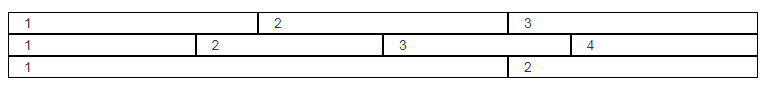
console.log('2 + "apple" = ' + iife.add(2, 'apple')); // Invalid Input

1. **Write the JQuery Equivalent for the following.**

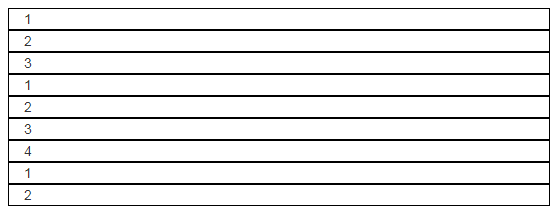
|  |  |
| --- | --- |
| **JQuery Equivalent** | **JavaScript Functions** |
| //Not sure | document.addEventListener("DOMContentLoaded", function () {}); |
| $(“.mySelector”) | document.getElementsByClassName("mySelector"); |
| $(“#mySelector”) | document.getElementById("mySelector"); |
| $(“p”) | document.getElementsByTagName(p); |
| $(“#myElement”).addClass(“myClass”) | document.getElementById("myElement").className = "myClass"; |
| $("#myElement").css("color", "#c00");  $("#myElement").css("backgroundColor" , "#eee");  $("#myElement").css("width" , "200px");  $("#myElement").css("height" , "100px");  $("#myElement").css("border" : "solid 2px red"); | var m = document.getElementById("myElement"),  c = m.style;  c.color = "#c00";  c.backgroundColor = "#eee";  c.width = "200px";  c.height = "100px";  c.border = "solid 2px red"; |
|  |  |
|  |  |

1. **Using only HTML with bootstrap classes create a grid that has three rows, which stack when viewport gets to 767px. First row should contain three equal cells. Second row should contain four equal cells. Third Row should contain two cells, one taking 2/3 of the available space, the other 1/3.**

**Desktop View**

****

**Mobile View**



I believe all I would have to do is set different “col” class and bootstraps built in media queries would automatically take care of their grid orientation. So on a desktop you could use the “col-lg” (or maybe “col-xl”?) and for mobile you could use “col-sm” (or maybe “col-xs”?). I have used bootstrap for 3 years now and their grid is my favorite feature.

1. **Explain the concept of Semantic Tagging in HTML5.**

Semantic Tagging makes it more clear what an HTML elements is used for. An example of this would be a <table> tag. You can easily tell that a table’s contents will be defined between that tag. A <div> is not a semantic tag because it gives no indication what contents will be inside of it.

1. **What meta tag is needed to optimize a web sites mobile presentation? Provide example of the tag and what it does.**

I am not sure about this. I’ve always used a CSS framework that abstracts media queries. As explained above you can use different bootstrap classes that indicate how elements should display based on window dimensions.

1. **Explain the matching set for the following CSS selector.**

a[data-dist^="yards"] > span

Not sure about this one

1. **Define the “box-model” and what effect “box-sizing” has on it.**

All html elements naturally take the form of a rectangle. So when creating a layout using html, you need to layout a screen as a page full of different rectangles. Box sizing would be the layout and spacing of those rectangles. So you can use margins to create space outside of an element’s rectangle, and padding to create space between your element’s rectangle and the content inside.

1. **What is the difference between block and inline-block elements?**

The “block” class displays elements in their own row. So if you put 2 blocks together in html code, on the screen they would stack vertically. Inline-block would display 2 blocks horizontally in the same row.

1. **What color would the text within the p element be based on the html & CSS below?**

<style>

.myElementClass p.myParagraphClass {

color: purple

}

p#myParagraphID .mySpanClass {

color: blue;

}

div.myElementClass.alternative p.myParagraphClass span.mySpanClass {

color: orange;

}

div#myElementID p.myParagraphClass {

color: red;

}

div.myElementClass.alternative p.myParagraphClass {

color: green;

}

</style>

<div id="myElementID" class="myElementClass alternative">

<p id="myParagraphID" class="myParagraphClass">

<span id="mySpanID" class="mySpanClass">Paragraph Content</span>

</p>

</div>

The paragraph would be blue because css selectors would give precedence to a reference to an ID over a reference to a class. So since the <span>’s parent is referred to by ID, and not by class like the orange definition, the element would be blue. If the second style definition was removed, the element would be orange because that is the only other definition that references the <span> by class.

1. **Difference between:**

**div > p**

And

**div + p**

If you use an IDE that generates html the first line would create a “div” with a “p” inside of it:

<div> <p> </p> </div>. The second line would create a “div” and a “p” after it: <div> </div> <p> </p>