FINAL ASSESSMENT RUBRIC

Guidelines for Grading

Below is an answer sheet for the final. Please note, when it comes to syntax/spelling/grammar, the answer should be marked wrong if the code will not compile or execute properly.

HTML / CSS

1. Create a CSS selector that selects all paragraph tags nested in elements with a class of “subscription-info” that is nested in an element with a class of ‘subscription’. *[1pt]*
   1. .subscription .subscription-info p { }
2. Taking the box model into account: If an element has a width of 200 pixels, padding of 5 pixels, a 1px border, and margin of 10 pixels. What is the total width of the element. *[1pt]*
   1. 232px
3. Create a CSS selector for an anchor element that will only apply its style when the user’s mouse passes over element. *[1pt]*
   1. *a:hover { }*
4. Explain the difference between block, inline, and inline block. *[1pt]*
   1. Block elements are elements that take up the entire line(or width) of the page. Inline elements are elements that only take up the width required to render the element. Inline-block is an inline element but you can specify height and width.

JAVASCRIPT

1. Declare and initialize the variable **limit** to be 25. Construct an if statement check if the variable limit is above or equal to 21. If true, the script should log the message, ‘limit is met or exceeded’. *[1pt]*

var limit = 25;

if(limit >= 21) {

console.log(“Limit is met or exceeded”);

} else {

console.log(“Limit is not met”);

}

1. Create a for loop that logs the integers from 1 to 50 to the console. For each integer that is evenly divisible by 10, log “Success” to the console. *[1pt]*

for(var i = 1; i < 51; i++) {

if(i % 10 === 0) {

console.log(‘Success’);

} else {

console.log(i);

}

}

1. Create a while loop that will prompt the user to enter their favorite food until the user types “Tacos”. *[1pt]*

var userGuess = prompt(“What is your favorite food?”);

while(userGuess !== “Tacos”) {

userGuess = prompt(“What is your favorite food?”);

}

1. Declare a function called findSum() that accepts two parameters. Inside the body of the function, return the value of both parameters added together. Then call the function with the arguments: 3 and 7. *[1pt]*

function findSum(parameter1, parameter2) {

return parameter1 + parameter2;

}

findSum(3, 7);

1. Declare a variable called catalog, initialize it as an array of objects. Each object should have properties of productName, description and unitPrice. Add an object for each of these catalog items: *[1pt]*

var catalog = [

{ productName: ‘Lamp’, description: ‘Standing lamp’, unitPrice: 8.73 },

{ productName: ‘Chair’, description: ‘What you sit in’, unitPrice: 66.35 },

{ productName: ‘Paperweight’, description: ‘For holding things down’, unitPrice: 3.46 }

];

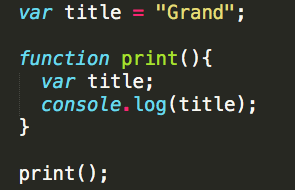
|  |  |  |
| --- | --- | --- |
| Product Name | Description | Price per unit |
| “Lamp” | “Standing lamp.” | 8.73 |
| “Chair” | “What you sit in.” | 66.35 |
| “Paperweight” | “For holding things down.” | 3.46 |

1. catalog.forEach(function(item) {  
    console.log(item.unitPrice);

});

1. Consider the following code, what will be printed to the log? In a few sentences, explain why. *[1pt]*

It will print undefined. The variable title is being re-declared inside the print function with no value. The variable title, within the print function, is undefined.



JQUERY

1. Using jQuery, select an element with an id of submit-button. Add an event handler using the on method. When the element is clicked, log the text of the button to the console. *[1pts]*

$(“submit-button”).on(“click”, function() {

alert(“clicked!”);

});

1. Using jQuery’s $.get() function, make a GET request to the url “http://example.com/json”. Log the data returned from the request to the console. *[1pt]*

$.get(“<http://example.com/json>”).done(function(data) {

console.log(data);

});

$.get(“<http://example.com/json>”, function(data) {

console.log(data);

});

ANGULAR

1. MVC - model view controller. The model is the information that is contained within the application. The view is what the user sees and interacts with. The controller is the business logic that processes information to update the model(which in turn, updates the view).
2. Using Angular, make an API call to http://api.example.com/florals and assign the response to a variable called florals. *[1pts total]*

$http({ url: “http://api.example.com/florals” , method: “GET” })

.then(function(response) {

florals = response;

});

1. Create an angular component called customerDetails *[1pt]*
   1. Set the templateUrl property to “path/to/my/component.html”.
   2. Set the controller property to a function that logs “Customer Details” to the console.

Write the HTML that would add this component to a view.

app.component(“customerDetails”, customerDetails);

var customerDetails = {

templateUrl: “path/to/my/component.html”,

controller: function() {

console.log(“Customer details”);

}

}

<customer-details></customer-details>

<customer:details></customer:details>

NODE.JS

1. Declare a variable http to require the http module. *[1pt]*

var http = require(‘http’);

1. In a file called animal-inventory.js is the following code…  
    var animals = [ "cow", "chicken", "sheep", "goat", "duck" ];  
    function printAnimals() {  
    animals.forEach(function(animal) {  
    console.log(‘We have a ‘ + animal);

});

}  
module.exports.animals = animals;  
module.exports.printAnimals = printAnimals;

Write the code required to import the content of animal-inventory.js and call the printAnimals function. *[2pts total]*

var animalStuff = require(“./animal-inventory”); -- rewards 1 pt

animalStuff.printAnimals(); -- rewards 1 pt

1. What command is required to run a script file through NodeJS? *[1pt]*

node server.js