|  |
| --- |
| **Part 0: Review Study Guide #1: Midterm! Then do this…** |

**Object:** An instantiated version of a class.

**class:** A “blueprint” of an object, where properties and methods are declared.

**constructor:** A special method that exists within a class to “construct” the class into a newly realized object.

**property:** A characteristic of an object/property that belongs to an object.

**method:** A body/snippet of code that exists within an object for use by the object.

**array:** Used to store multiple bits of data in a single variable.

**array index:** Used to retrieve data from a specific part of the list.

e.g. myArray = [“summer”, “spring”, “winter”, “fall”]

myArray[0] = “summer”, myArray[myArray.length - 1] = “fall”, myArray[2] = “winter”

**finite state machine:** Logic to store the changes in state based on a trigger or change in property.

**null:** A lack of anything, but still ‘defined’.

1. Which of the following are *invalid (illegal)* JavaScript variable identifiers? Why?

a. RESULT -- constant

b**. Factorial!**

c. MAX\_SIZE – underscores are ok

d**. black&white**

e. answer\_7

***no special characters are allowed.***

2. Which of the following are *not* *appropriately* *named* JavaScript variable identifiers? Why?

**a. qqq**

b. totalValue

**c. theNextValueInTheVerySpecialList**

d. player1Score

**A: does not describe the variable at all**

**C: too long to be a useful variable name**

3. What is the output produced by the following code statement?

console.log("50 plus 25 is " + (50 + 25));

* **50 plus 25 is 75**

4. The function calcDamage() returns a value of 34. Is the following code snippet valid? Why or why not?

let health = 250;

health -= calcDamage();

* This code snippet **is valid** as it declares a variable *health* and then reduces its value by 34, a number which is returned the function *calcDamage();*

5. What is the output produced by the following code block?

let state = "VA";

switch(state) {

case "AK":

console.log("Alaska");

break;

case "VA":

console.log("Virginia");

default:

console.log("could be a contender");

}

* Virginia
* could be a contender

Because there is a missing *break*; statement after the *console.log(“Virginia”);* it will continue to execute the following cases.

6. What is the length of a 1D array with indexes from 0 to 25?

The length of a 1D array with indexes from 0-25 will be: **26**

*Include 0 as an index.*

***IF*** *there was only 25 element in the array, the indexes would be 0-24*

***Write the code necessary for each question.***

7. Given an array with the identifier myColors, iterate through the array using a loop. Inside the loop, print the values of every element in the array to the console window.

let myColors = [“blue”, “pink”, “yellow”, “brown”];

**for(let i = 0; i < myColors.length; i++) { // THIS IS GOOD**

**console.log(myColors[i]);**

**}**

**for(let i = 0; i < 4; i++) { // THIS IS WORSE**

**console.log(myColors[i]);**

**}**

8. Write the JavaScript code that creates an array and assigns values to the array. The array name should be precipitation and the values are 0.45, 2.78, 0.04 and 1.22.

**let precipitation = [0.45, 2.78, 0.04, 1.22]**

9. Complete the for loop to assign values into the ascendingNumbers array. When the loop is finished running, the array should hold the values 1, 2, 3, 4, 5.

let ascendingNumbers = [];

**for(let I = 1; I < 6; I++){ // I is capital because word is annoying**

**ascendingNumbers.push(I); // Adds to the end of the list**

**}**

**for(let I = 1; I < 6; I++){ // I is capital because word is annoying**

**ascendingNumbers[I - 1] = I; // Direct array index, in this case it’s a little bit gross**

**}**

10. Use the Cake class to answer this question.

class Cake {

constructor(flavor, price) {

this.flavor = flavor;

this.price = price;

}

display() {

console.log(this.flavor, "cake, $" +

this.price);

}

}

1. Declare and initialize an array.
2. Add the following new Cakes to the array: "Vanilla", 10.99. "Raspberry", 14.99. "Chocolate", 12.99.
3. Using their display() methods, print information about all Cakes in the array.

**// a**

let cakes = []

**// b**

cakes.push(new Cake(“Vanilla”, 10.99));

cakes.push(new Cake(“Raspberry”, 14.99));

cakes.push(new Cake(“Chocolate”, 12.99));

**// c**

for(let i = 0; i < cakes.length; i++) {

cakes[i].display();

}

***Short answer.***

11. Briefly describe the array methods, properties, etc. below:

* 1. .pop()

Takes last element of array out of the array

* 1. .push()

Adds a new variable to the end of the array

* 1. [<index>]

Represents a specific element at position <index> in an array

* 1. .length

The length of the array. (Don’t forget arrays start from 0)

12. What is the value printed for ball1.x?

let ball1 = new Ball();

ball1.x = 1;

**let ball2 = ball1;** // 🡨 Highlight for Q14

ball2.x = 3;

console.log(ball1.x); // 🡨 what value is printed?

* **3 // REMEMBER PASS BY VALUE/REFERENCE**

13. Explain what happened in Q#12’s highlighted line of code.

**The address of ball1 was assigned to ball2, meaning now we have two pointers to the same object, so whatever we do to one object will be reflected by both variables.**

* **REMEMBER PASS BY VALUE/REFERENCE**
* <https://www.mathwarehouse.com/programming/passing-by-value-vs-by-reference-visual-explanation.php>

*Given the code for the Snowball class, answer the questions below.*

class Snowball {

*constructor*(diameter) {

this.diameter = diameter;

this.x = random(80, width – 80);

this.y = random(80, height – 80);

}

melt() {

this.diameter = this.diameter – 0.5;

}

display() {

circle(this.x, this.y, this.diameter);

}

}

14. Write the line of code that would be inside setup() that creates a Snowball object with the name smalland a diameter of 20.

=> let small = new Snowball(20);

15. Write the line of setup() code that accesses the small Snowball’s diameter to print to the console window.

=> console.log(small.diameter);

16. Write the line of setup() code that calls the melt() method on the small Snowball.

=> small.melt();

17. Complete the following code to create an array named projectiles*.* Fill it with 5 new Snowball instances, each having a starting diameter of 30.

let projectiles = []

for( let i = 0; i < 5; i++ ) {

projectiles.push(new Snowball(30));

}

18. The snow melted a bit on a warm day, then melted again the next day. Write the code to access the Snowball object in the array at index 3, melt it twice, and display it in the canvas.

**projectiles[3].melt();**

**projectiles[3].melt();**

**Projectiles[3].display();**

19. What is the purpose of the following setup() code?

textSize(32);

textFont("Tahoma");

* To establish a default textsize and font

20. Write a splatImages() function. It accepts one parameter - an array of image objects. It should draw every image in the array at a random x-y location within the canvas (don't worry about image edges going partially beyond the canvas).

**function splatImages(array) {**

**for(let i = 0; i < array.length; i++) {**

**image(array[i], random(width), random(height));**

**}**

**}**

21. Code a method updateState() that contains a conditional block that updates this.state based on the current value of this.state and this.daysWater (days since last watered) according to the state transition table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instance state** | .daysWater < 5 | 5 <= .daysWater < 10 | 10 <= .daysWater < 15 |
| "dry" | "healthy" | "dry" | "dry" |
| "healthy" | "drowning" | "healthy" | "dry" |
| "drowning" | "drowning" | "drowning" | "healthy" |

**class ??? {**

**…**

**updateState() {**

**if(this.daysWater < 5 && this.state === “dry”) {**

**this.state = “healthy”;**

**} else if(this.daysWater < 5 && this.state === “healthy”) {**

**this.state = “drowning”;**

**} else if(this.daysWater < 15**

**&& this.daysWater >= 10**

**&& this.state === “healthy”) {**

**this.state = “dry”**

**} else if(this.daysWater < 15**

**&& this.daysWater >= 10**

**&& this.state === “drowning”) {**

**this.state = “healthy”**

**}**

**}**

**…**

**}**

22. Strings are concatenated by using the \_\_\_ operator.

\* - .plus() **+**

23. The special character ‘\n’ creates a \_\_\_ in the string’s formatting.

**new line** tab escape shift

24. Write setup() code to create a p5 radio object named bestSummer (assume it has already been declared as a global variable). The options should be “June”, “July”, and “August”.

**function setup() {**

**bestSummer = createRadio();**

**bestSummer.option(“June”);**

**bestSummer.option( “July”);**

**bestSummer.option( “August”);**

**}**

25. Write draw() code that will set the canvas background color according to the current .value() of bestSummer:

* June: “pink”
* July: “orange”
* August: “red”
* (nothing chosen): “gray”

**function draw() {**

**if(bestSummer.value() === “June”) {**

**background(“pink”);**

**} else if(bestSummer.value() === “July”) {**

**background(“orange”);**

**} else if(bestSummer.value() === “August”) {**

**background(“red”);**

**} else {**

**Background(“gray”);**

**}**

**}**

OR YOU CAN DO

**function draw() {**

**switch(bestSummer.value())**

**background(“pink”);**

**break;**

**case: “July”**

**background(“orange”);**

**break;**

**case: “August”**

**background(“red”);**

**break;**

**default:**

**background(“gray”**

**break;**

**}**

27. Write a function named isEvenSum() that accepts two numbers as parameters, and returns true if the sum of the numbers is even. If the sum is odd, it returns false.

**function isEvenSum(num1, num2) {**

**let sum = num1 + num2;**

**if(sum % 2 === 0) {**

**return true;**

**} else {**

**return false;**

**}**

**}**

28.Write code in a preload() function that will load the image stored externally in media/octavia.png and store it in the global variable, octaviaSmiles.

**let octaviaSmiles;**

**function preload() {**

**octaviaSmiles = loadImage(“media/octavia.png”)**

**}**

29. Write code in a mousePressed() function that will draw the image in octaviaSmiles on the canvas at the mouse location.

**function mousePressed() {**

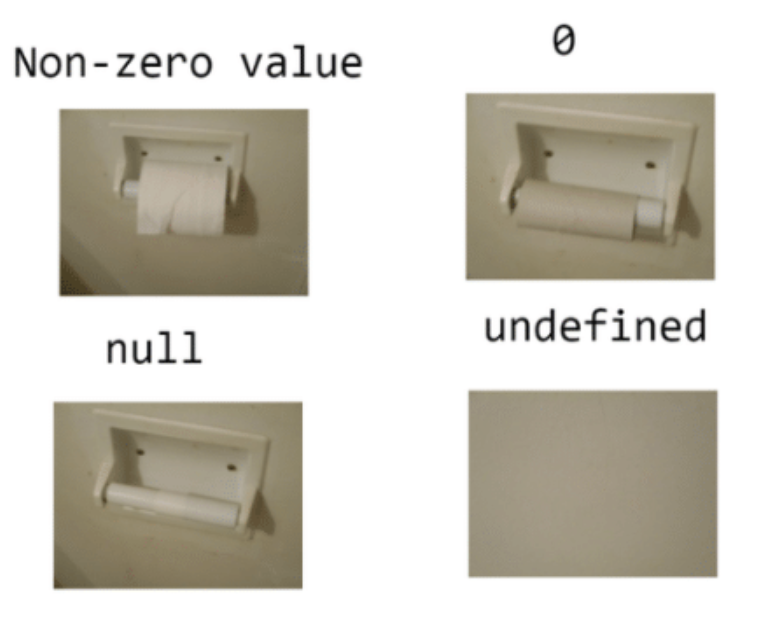
**image(octaviaSmiles, mouseX, mouseY);**

**}**

30. What is the difference between undefined and null?

Undefined means not even the variable has been declared, there is nothing there.

Null means the variable exists but has no value.



31. Assume there is a Word class whose constructor() expects a label string and x and y coordinates as parameters, i.e. constructor(label, x, y). Finish this setup() function to fill words[] with a 2D array of new Word instances based on the local 2D array named labels.

* Each x value for a Word in a row should be wordWidth apart.
* Each y value for a row of Words should wordHeight apart.

//Global variable

let words = []; //stores 2D array of Word objects

setup() {

let labels = [["snowy vistas", "ice forts", "warm blankets"],  
["fire crackling", "scarf", "gloves"],  
["hot cocoa", "long, cold nights", "ice hockey"],  
["sledding", "latkes", "chili"]];

let wordWidth = 200;

let wordHeight = 60;

createCanvas(600, 240);

// Fill 2D array of Word objects based on the 2D array labels

**for(let i = 0; i < labels.length; i++) {**

**let innerArray = []**

**let currentY = i \* wordHeight;**

**for(let j = 0; j < labels[i].length; j++) {**

**let currentX = j \* wordWidth;**

**innerArray[j] = new Word(labels[i][j], currentX, currentY);**

**}**

**words[i] = innerArray;**

**}**

}

Common Coding Errors: Comment what is wrong with the following code snippets:

32.

let i = 0;

if(**i = 8**) { // Should be **==** OR **===**

console.log(“i is 8!”);

}

33.

if(**keyIsDOWN**(65)) {

*// There is no function keyIsDOWN, should be keyIsDown*

console.log(“A is pressed”);

}

34.

let i = 8;

if(i === 0)**;** { *// THERE IS A SEMICOLON AFTER THE IF STATEMENT.*

*// This one is subtle, keep an eye out.*

console.log(“i is 0!”);

}

35.

if(**keyPressed**) { *// keyPressed is a p5 function, not a Boolean*

*// keyIsPressed is the Boolean you are looking for.*

console.log(“A key has been pressed!”);

}

// keyIsPressed: <https://p5js.org/reference/#/p5/keyIsPressed>

// keyPressed: <https://p5js.org/reference/#/p5/keyPressed>

36.

function mouseIsPressed() { // Same as above, but reversed, and mouse

console.log(“mouse is clicked”);

}

37.

class MyClass {

**constucter**(x, y) { // Spelled wrong, should be constructor

this.x = x;

this.y = y;

}

display() {

fill(“blue”);

rect(this.x, this.y, 400, 400);

}

}

38.

class MyClass2 {

constructor(color, size) {

this.color = color;

this.size = size;

}

display() {

fill(color); *// missing this.*

rect(30,30,size,size); *// missing this.*

}

}

39.

if(keyIsDown(**‘P’**)) { // keyIsDown is not looking for a char

// It’s looking for the keycode (in the case of ‘P’ its: 80)

console.log(“P is pressed!”);

}