Grant questions.

1. Is there a video about the visual studio that gives you ideas on short cuts etc.
2. Is there a way to program your keyboard to have < > and { } to be used first before the . or [
3. How do I slow down from what I already know to what I am trying to learn. I feel myself jumping to far forward and asking to many questions.
4. What does the CSS box model do?
5. I am not able to run my javascript to check my work.

Sources of info:

<https://developer.mozilla.org/en-US/docs/Web/HTML>

<!DOCTYPE HTML>

**<HTML>**

<head>

<title> This is a sample html page</title>

</head>

<body>This is the body of my sample html page</body>

<h1>This the my example of Headers<h1> Takes up the entire width of the page  
<h2> This the my example of Headers <h2>  
<h3> This the my example of Headers <h3>  
<h4> This the my example of Headers <h4>  
<h5> This the my example of Headers <h5>  
<h6> This the my example of Headers <h6>

**SPAN**The <span> tag is used to group inline-elements in a document. ommonly used to identify smaller groupings of text within a block-level element.  
The <span> tag provides no visual change by itself.  
The <span> tag provides a way to add a hook to a part of a text or a part of a document.

<span>Hello </span> Span only takes up the space required for content.  
<span>World! </span>  
<span> makes more sense on CSS than straight HTML</span>

<span style="background-color:powderblue;">

       HIGHLIGHT BLUE</span>

**Image TAG**

<img scr=” ” /> link to an image. How do you define the size of the image and not make it blurry. **Self-closing tag.**

**DIV**  
The <div> tag defines a division or a section in an HTML document.  
The <div> element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.

**LINK for CSS**

<link href="style.css"/> Link for CSS  
  
**LINK for SCRIPT**  
<script src="app.js"></script> Link to javascript

Comments in HTML

<!--This is a comment. Comments are not displayed in the browser-->

**<form>**

<input /> **another self-closing tag. Take inform from the user.**

</form>

**<Button**>Click Me</Button>

<a href= “www.google.com “> Go to google</a>

**LISTS**

**<ul> Unordered list: Should be BULLET POINTS, Here is the location of the** [**list of different property styles of bullet points**](https://www.w3schools.com/cssref/pr_list-style-type.asp)**.**

<li>Item 1 </li> li=List item  
<li>Item 2 </li>

</ul>

**<ol> ordered list = 1. 2. 3.**

<li>Item 1 </li>  
<li>Item 2 </li>

</ol>

You can use roman numerals by using <ol type=”i”> </ol> used for close

You can also used nested lists.

**TABLES ARE NOT A GOOD CHOICE TO USE ON WEBSITES. THEY CAN GET VERY DETAILED AND VERY CONFUSING VERY FAST**

|  |  |
| --- | --- |
| <table> Creates the table |  |
| <tr> Table Row Table Row |
| <td> iOS Development</td> Table Cell  <td>©©©©©</td> **These are supposed to be stars**  <td>Web Development</td> Table Cell  <td>©©©©©</td> **These are supposed to be stars**  <td> Photography</td> Table Cell  <td>©©</td> **These are supposed to be stars**  </tr> Close the Row  </table> Closes the table |
| In order to make columns  You need to create  <table> Table One  <tr> Create Row  <td>One</td> Create Cell  <td>Two</td> Create Cell  </tr> Close Row  </table> Close Table  <table> Create second Column  <tr> Create Row  <td>Three</td> Create Cell  <td>Four</td> Create Cell  </tr> Close Row  </table> Close Table |  |

<Form>

<label>Name:</label>

<input type=”text”>  
<input type=”submit”>  
<input type=”file”>  
<input type=”date”>  
<input type=”radio”>  
<input type=”range”>

<Input />

</Form>

<Button> Click me </Button>

<a href=”www.google.com”> Go to Google </a>

</html>

**CSS Cascading Style Sheets**

**This is a page that you can get codes for colors** [**colorhunt.co**](file:///C:\Users\jeff_\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\colorhunt.co) **for colors, you can use the name, #45154 or RGB**

Selector=Who? { Property= What? : value=How;}

|  |  |
| --- | --- |
|  |  |

You can have more than one rule to a selector, and you can have a numerous selectors. IE

**Note: 2 selectors are being used for the same code:** H1, H2  
 {  
color: green;   
font-size: 200px;  
}

**NOTE: TWO Best practice rules:**

1. **Put the properties in alphabetical order**
2. **Put the Property: Value; on a separate line so you they are easier to read, and you can catch mistakes easier**

**Comments in CSS**

/\* ……..\*/

**CSS Stylesheets <link rel=”stylesheet” href=”style.css”>**

Here is how to figure out how to choose and write a selector.

* Elements = selector
  + Child elements
* Class= .selector **MOST COMMONLY USED**
  + Child classes
* ID= #selector **TRY TO STAY AWAY FROM ID’S IN CSS, JAVACRIPT MAINLY**
  + Child ID

 In CSS **the asterisk, \*,** is the ***universal selector,*** which selects every element.

To reference only the parent here is an example of how to do that…..

**HTML**

<div class-“parent”>  
 <p>Dark Coffee</p>  
 <p>Medium Coffee</p>  
 <div>

<p> Medium-Light Coffee</p>

</div>

<p> Light Coffee </p>

**CSS**

.parent > p {  
 background-color: green;

}

**HTML**

Or you can combine selectors.

<div class=”hotdog”>

<p> …</p>

<p> …</p>

<p class=”mustard”> …</p>

</div>

**CSS**

.hotdog p {

background: brown;

}

.hotdog p.mustard {

background: yellow;

}

These are two great reference for writing [CSS selectors](https://www.w3schools.com/cssref/css_selectors.asp)

On this site you will find a list of all the [keywords for CSS not javascript.](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference#Keyword_index)

**ELEMENTS** are typed as is…Examples

* Body { property; value; }
* Li
* Img
* P
* H1
* P, H1 : can have the same rules

**CLASS** starts with a “ . ” period For Example:

on HTML page <div class= “mentor”>

on CSS page  **.mentor** { property: value;}

**THESE ARE THE MOST COMMON.**

**ID** starts with a # “pound”.

#ID selectors are rigid and don't allow for reuse. If possible, first try to use a tag name, one of the new HTML5 elements, or even a pseudo-class. **TRY AND STAY WAY FROM THESE**

# ID **‘**selector name**’** {  
 color; red;

}

The asterisk symbol will target every single element on the page. For example.

\*{

Margin:0;  
Padding:0;

}

The **ASTERISK “ \* ” SYMBOL WILL TARGET EVERY SINGLE ELEMENT ON THE PAGE**. IT IS NOT RECOMMENDED TO USE THIS WAY. USE ON CHILD SELECTORS ONLY!

#container \* {

Border: 1px solid black;

}

CSS requires that you close the code with the “ **;** “ semi-colon if not then the code will not work.

Property=Position: Fixed;

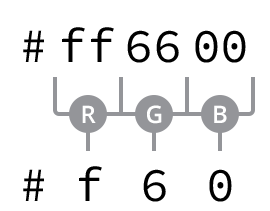
**Colors**

**Best resource for fonts and backgrounds**

* [**Adobe**](file:///C:\Users\jeff_\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\color.adobe.com)
* [**Colorhunt**](file:///C:\Users\jeff_\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\colorhunt.co)

Three ways to define colors

1. Hexadecimal #A4F6B9
2. Similar Hexadecimal #A4B
3. Functional Notation (RGB) = in CSS written like .’selector’ { rgb (R,G,B)
   1. These are all RGB values which are 0 thru 255
   2. If you need to add opacity .’selector’ { rgba (55,1555,255,.1)
      1. The values of a are zero thru one so .01 to 1



**Fonts**

Great site for looking for fonts [Google Fonts](https://fonts.google.com)

Great site for choosing the right fonts [Designbold](https://www.designbold.com/academy/en/choosing-the-right-fonts-for-website/)

**Display Property**

Block Elements that take up the entire width of the webpage  
In-line Elements that take up to as much width as they need.

In HTML lists always are placed in block form if you want to make it inline then….

In CSS to use inline display use the name of the selector that want to inline and write

‘selector’ {

Display: inline; **NOTE:** **top and bottom margins & paddings are not respected**

}

In HTML images are always inline with one another in CSS to make it a block form

‘selector’ (

Display: block;

}

‘

Inline block for a [Great Reference](https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_span1)

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

The following example shows the different behavior of display: inline, display: inline-block and display: block:

‘selector’ {

Display: inline-block;

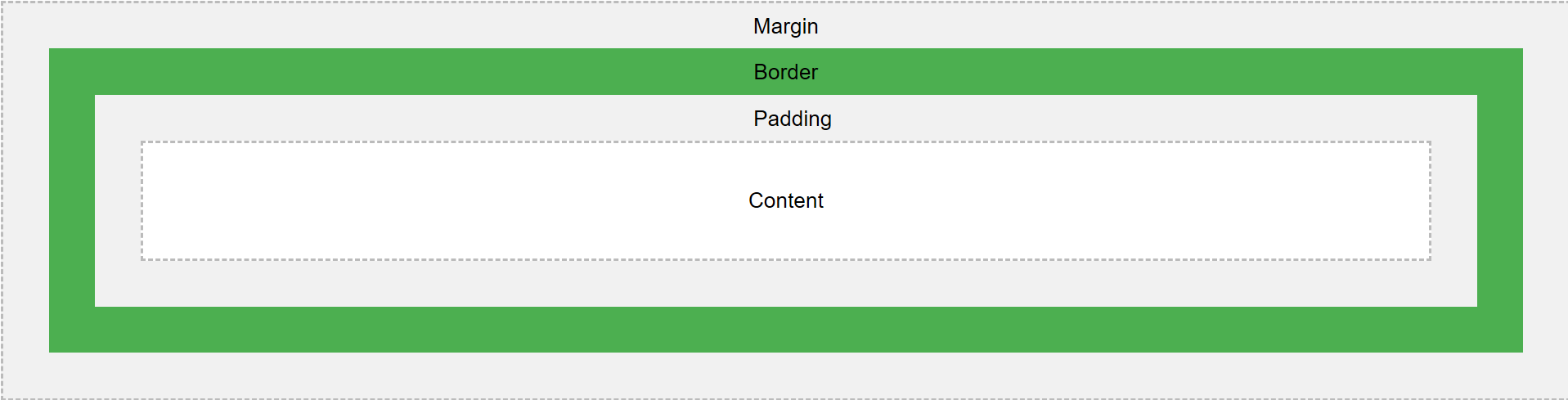
}

**More indepth level inline block**

‘selector’ {  
  display: inline; /\* the default for span \*/  
  width: 100px;  
  height: 100px;  
  padding: 5px;  
  border: 1px solid blue;   
  background-color: yellow;   
}

**Box Model**

* **Content** – The text, image, or other media content in the element. **Sets the area the content “text, image, etc” will be held.**
* **Padding** – The space between the box’s content and its border. **Will decrease the content space and pad the font, giving the font a smaller area. Moving the content around**
* **Border** – The line between the box’s padding and margin. **Can make the area around the content box bigger or smaller but doesn’t effect the size of content area.**
* **Margin** – The space between the box and surrounding boxes. **Effects the location of where the box is placed on the website large margins will put the box more towards the middle of the website.**



\* {

box-sizing: border-box;

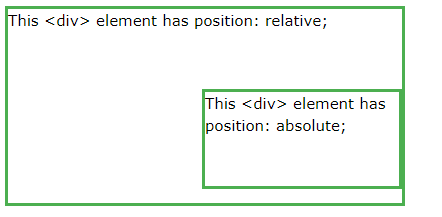
}

**Make sure this goes in each CSS sheet you are working with Box Modelling**

Box-sizing Defines how the width and height of an element are calculated: should they include padding and borders, or not.

**CSS Positions**

* Static- HTML elements are positioned static by default.
* Relative- is like block formatting, all elements will go below it.
  + Left:
  + Right:
* Absolute- you tell where the cell to be inside the relative cell
  + Height:
  + Width:
  + Left:
  + Right:
  + Top:
  + Bottom:



* Fixed- to the viewable area no matter where you move on the webpage. IE a NavBar

**HTML**

<div class="nav-bar">

<span class="home">Home</span>

<span class="menu">Menu</span>

<span class="about">About</span>

</div>

**CSS**

.nav-bar {

position: fixed;

width: 100%;

margin: 10px;

background-color: rgb (55,44,22,.4);

}

.about {

position: relative;

left: 315px;

}

.menu {

position: relative;

left: 150px;

}

Javascript  
Data Types

| **Variable** | **Explanation** | **Example** |
| --- | --- | --- |
| [String](https://developer.mozilla.org/en-US/docs/Glossary/String) | A sequence of text known as a string. To signify that the value is a string, you must enclose it in quote marks. | let myVariable = 'Bob'; |
| [Number](https://developer.mozilla.org/en-US/docs/Glossary/Number) | A number. Numbers don't have quotes around them. | let myVariable = 10; |
| [Boolean](https://developer.mozilla.org/en-US/docs/Glossary/Boolean) | A True/False value. The words true and false are special keywords in JS, and don't need quotes. | let myVariable = true; |
| [Array](https://developer.mozilla.org/en-US/docs/Glossary/Array) | A structure that allows you to store multiple values in one single reference. | let myVariable = [1,'Bob','Steve',10]; Refer to each member of the array like this: myVariable[0], myVariable[1], etc. |
| [Object](https://developer.mozilla.org/en-US/docs/Glossary/Object) | Basically, anything. Everything in JavaScript is an object, and can be stored in a variable. Keep this in mind as you learn. | let myVariable = document.querySelector('h1'); All of the above examples too. |

Var age = 10 Var=Variable then you need to declare the variable in this case “10” so in the program anytime you use the word age it =10

Consule.log(age) if you then write the Console.log(age + age) it will = 20

You have two main data types

1. Primitive
   1. Strings var name=’joe’
   2. Numbers var age=20 Don’t use quotes
   3. Booleans = true or false
      1. Var isAlive = true (if you put in quotes this make it a string)
      2. hasDoneTaxes = false
2. Complex= a collection of data types
   1. Arrays = [1,2,3,4] List of similar items
      1. Var numbers= [ 1, 2, 3 ] need to be comma separated
      2. Numbers[0] indexing starts at zero
   2. Objects Explains a singular item in depth (key value pairs)
      1. Var person = {
      2. Name: “Steve”,
      3. Age: 21,
      4. hairColor: “brown”
      5. isAlive: true NOTE for very last item there is no comma.

|  |  |
| --- | --- |
| **Code** | **Output** |
| \’ | Single Quote |
| \” | Double Quote |
| \\ | Backslash |
| \n | Newline |
| \r | Carriage Return |
| \t | Tab |
| \b | Backspace |
| \f | Form Feed |

**Example** var myStr = “FirstLine\n\t\\SecondLine\nThirdLine”

**Example** Var ourStr = “I come first. “ + “I come second.”;

Var myStr = “This is the start. “ + “This is the end.”

Console.log(myStr);

**Example** var myStr = “This is the first sentence. “

myStr=+”This is the second sentence.“

console.log(myStr);

**Arrays**

Var myNumbers = [1, 2, 3, 4, 5]

0 1 2 3 4

Var colors = [“blue”, “green”, “red”, “yellow”, “purple”]

Console.log(colors[1])

Every Array has a length

Console.log( colors.length

Strings also have length property  
var name = “joe”

Console.log(name.length)= (name[1] = =

**Conditionals**

If(true) {

Console.log(It is true!”)

}

If(false) {

Console.log(It is true!”)

} else {

Console.log (It is false!”

If (2 === 2) {  
console.log(“Two is equal to two”)

} else {

Console.log(Two is not equal to two”)

}

If (2 === 4) {  
console.log(“Two is equal to two”)

} else {

Console.log(Two is not equal to four”)

}

If (2 === 4) {  
console.log(“Two is equal to two”)

} else {(2 === 5){

Console.log(Two is not equal to five”)

} else {(2 === 2){

Console.log(Two is not equal to two”)

}

If (2 === 4) {  
console.log(“Two is equal to two”)

} else {(2 === 5){

Console.log(Two is not equal to five”)

} else {(2 === 6){

Console.log(Two is not equal to six”)

} else {  
console.log(“I don’t know what is going on”)

}

**Operators Symbols**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| >, | <, | >=, | <=, | ==, | ===, | !=, | !== |
| Less than | Greater  than | Less than or equal to | greater than or equal to | Loosely equal to | Strictly equal to | Loosely not equal to | Strictly not equal to |

If (“2” == 2){  
console.log(“They are equal”)

} This would be true REMEMBER THAT “2” is a STRING so loosely it is equal to 2 as 2 matches 2. If it was === strict then it would be False.

There are Truthy and Falsey

All Falsey

* 0
* “ ”
* Null
* Undefined
* False
* NaN not a number

If(“”) {

console.log(“it is truthy”)

} else {

console.log(“it is falsey”)

}

If you were to put a character into “a” then it would be truthy

**Logic Operators**

&& - and

|| - or

! – not

If (2 === 2 && 2 === 3) {

console.log(It’s working!”)

} else {

console.log(“Its’s not working!”)

}

If (2 === 2 || 2 === 3) {

console.log(It’s working!”)

} else {

console.log(“Its’s not working!”)

}

If (2 === 2 && 2 !== 3) {

console.log(It’s working!”)

} else {

console.log(“Its’s not working!”)

}

**Order of Operations**

+ - \* /

Example If (2 === 2 && 3 === 3) && 2 !== 3){

console.log(It’s working!”)

} else {

console.log(“Its’s not working!”)

}

**Switch**

Here is a Switch Statement

Var color = “blue”

Switch(color) {  
 Case “red”:  
console.log(“The color is red”)  
**}** take out then add break this will stop the code each time

Case “blue”:  
console.log(“The color is blue”)  
break

Case “yellow”:  
console.log(“The color is yellow”)  
break

Default:

console.log(“The color is not red, blue, or yellow”)

}

console.log(

var person = “Bobby”;  
var age= “12”   
}

if (“age”<=18) {  
console.log(“ + person + “is old enough to go to this movie!”)  
} else {  
Console.log(“ + person + “is not old enough to go to this movie!”)

}

var person = “Bobby”;

var age= “12”

if (“age” <= 18) {

console.log(“ + person + “is old enough to go to this movie!”)

} else ("age" > 17) {

Console.log(“ + person + “is not old enough to go to this movie!”)

}

If (2 === 4) {  
console.log(“Two is equal to two”)

} else {

Console.log(Two is not equal to four”)

}

**Loops**

For(var i = 0; i < 100; i ++){

Console.log(i)

}

Var favfoods = [“pizza”’ “pasta”’ “ice cream”, “banana”]

favFoods.length //4

\Console.log(favFoods[o])

\Console.log(favFoods[1])

\Console.log(favFoods[2])

\Console.log(favFoods[3])

For(var i =0; I < favfoods.length; i++) {

Console.log( favFoods[i] ) **This is really important syntax**

}

Var numbers = [1, 2, 3, 4, 5, 6]  
numbers[0]

For(var i =0; I < numbers.length; i++) {

If (numbers[i] % 2 === 0) [

Console.log(numbers[i] )

}

Var count = 0

While(count <10) {  
console.log(‘hi’)

Count++

}

**Functions** DRY = Do Not Repeat Yourself

Statement is any line of code

Var name= “nate”

**(This is the function Declaration)**

Function sum() {

Console.log(2+2)

}

**(Function Expression)**

var mySumFunction = function () {

Console.log (2+2)

**(Parameters)**

Function sum(num1, num2) {

Return num1 + num2

}

//We are calling the function or execute the function

Var result =sum(10, 25) arguments

Console.log(result)

Function myFunc(data) {

Console.log(data)

}

Myfunc(1)

Myfunc(“hello”)

Myfunc(true)

Myfunc([1, 2, 3, 4])

Myfunc({name: “joe”})

Function loopThroughArr(array){

For (var i = 0; i < array.length; i++){

Console.log(array[i]

}

Objects best used to describe something in-depth

Key: value

Var person = {

Name: “Rick”

Age: 70

Friends: [ “Morty”, “Joe”, “Sam”, “Samantha”],

Address: {“123 street”,

City: “Somewhere”

}

Two notations for accessing an object’s data

* Dot “.”

Console.log(person.name)

Console.log(person.age)

Console.log(person.friends)

Console.log(person.address.street)

* Bracket notation [ ]

Console.log( person [“address”] [“city”]

Var car = {

Type: “Honda”  
make: “Civic”  
wheels: 4  
  
Honk: function(){  
Console.log( car [“Honk”]  
}

Car.honk()

//Car.hasHadAccident=true  
// Console.log( person (car)

**Objects can hold methods**

Var car = {

Type: “Honda”  
make: “Civic”  
wheels: 4  
honkSound: “Blleerrp”’  
Honk: function(){  
Console.log( this.honkSound ) As “THIS” represents “CAR”  
}

How are objects referenced

Objects and Arrays are passed by reference

Var othercar = car

**Conventions**

1. Casing= camelCased 1st word is lowercase secondword is uppercase
   1. Var lastName
   2. Var myLocationIsInUtah
2. White space
   1. For(var I = 0; I < 10; i++){ use an extra space so you can read the code more readily.
3. Naming
   1. Var fName = “Nate”
   2. Var lName = “Jensen”
   3. Array ‘s can be different
      1. var friends = []
      2. var isDone = true
      3. var isGameOver = false
      4. function sum(num1, num2){ }
4. Braces
   1. Hit enter to keep everything lined up the Visual Studio will intend everything for you.
      1. Var person ={ enter
5. Tab (Indenting)
   1. If (2 ===2) {

Console.log(“they are equal”)

If(3 ===3) {

Console.log(“they are equal”)

}  
}

1. HTML Tags
   1. Use lower case <div>
2. Comments
   1. //
3. Unix casing
   1. Lower case and naming conventions, silly\_cia

**What is a Dom? Document Object Model**

**Javascript object that represents the document or webpage, it provides a set of tools for the developer to use to manipulate the web page.**

**Index.js**

**document**

**Event Listener**

**The Standard Loop for running conditional statement with math**

**For (initializer; exit condition; final-expression0 {**

**// code to run**

**}**

**Counter, which is initialized with a certain value- this is the starting point of the loop. (“Start: I have no food, above).**

**Exit condition, which is the under which the loop stops- usually the counter reaching a certain value.**

**An iterator, which is generally increments the counter by a small amount on each successive loop**

**Fr**

**In the for statement**

**Var is what initiates the condition**

1. **String: “ “,**
2. **Number 10,**
3. **Boolean**

**Complex**

1. **Array:** [1,2,3,4] List of similar items
   1. String Array is [“Bob”, “Rick”, “Jack”, “Tom”]
   2. Numbered Array [ 1, 2, 3 ] need to be comma separated
      1. To start an array you must note that it always starts [0] indexing starts at zero, so if you have 1, 2, 3, 4 then there are how many arrays [3] always subtract 1
2. **Objects**: Explains a singular item in depth (key value pairs)
   1. Var person = {
      1. Name: “Steve”, (String)
      2. Age: 21, (number)
      3. hairColor: “brown” (String)

**}**