Painted Brain's CodiePie Workshop Curriculum

BASIC

8 Week Course

2018

Authors: Vincent Gomez and David Israelian

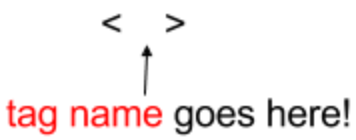
Coding Workshop: Basic

Table of Contents
(Include HTML, CSS, and Javascript)

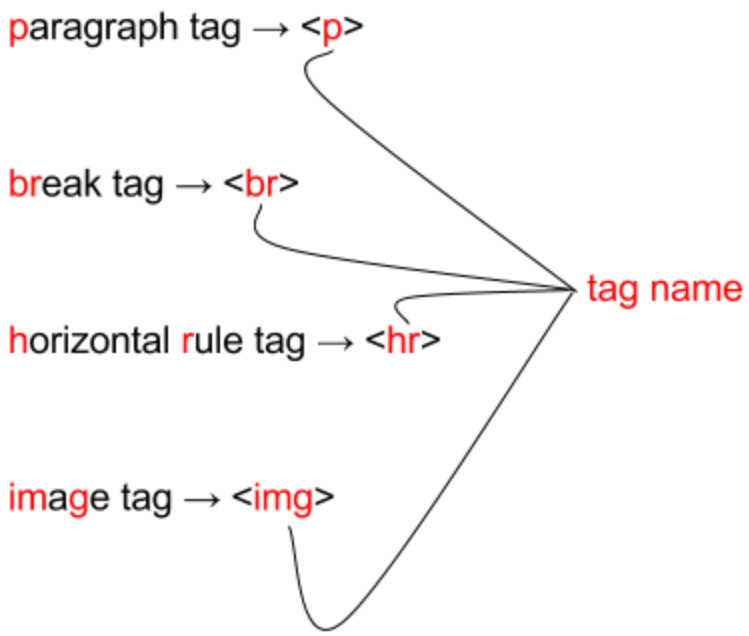
- Section 1 - Getting Started!
    - What is coding?
        - Brief history of HTML and definition
    - Download text editor
        - TextWrangler, Sublime, Atom
    - Access browser console
    - Edit/Save Files
    - Pair-coding- working with another coder makes both of your stronger coders!
        - Share knowledge
        - Different points of views
        - Practice communicating and explaining your ideas
    - “Google-fu”
        - The ability to research for answers to any questions you may have about coding.
    - Resources
        - https://www.w3schools.com - a useful handbook of practically everything you need to know.

        - https://www.youtube.com/user/thenewboston - youtube tutorials that cover web development in plain English.
        - https://www.codecademy.com - online curriculum with lessons and exercises.
        - https://www.freecodecamp.com - online curriculum with lessons, exercises, and projects.
        - http://codepen.io - code and see changes to your code immediately

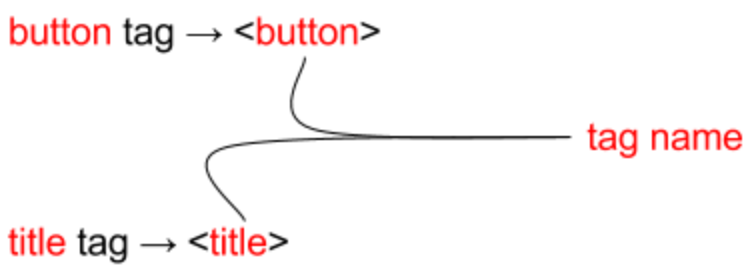
- Section 2 - HTML Basics/Syntax- play close attention more to HOW HTML is written.
    - Tags are written with a tag name between these two symbols “<” and “>”. It looks like this:



Usually, each tag name is an abbreviation of what its full name is:



There are a few that use its full name though. Examples:



A “<” without the “>” is NOT a tag.

- There are two types of tags:
  - Tags that can be written alone. Examples:
    - <!DOCTYPE HTML> - used to declare a document as an html document.
    -
 - adds a line break.

- <hr> - draws a horizontal line.
- <img> - puts an image in the website.

- Tags that require closing tags. A closing tag is just like an opening tag except there is an extra "/" symbol before the tag name. Between the opening and closing tag will be stuff. We'll look more at this stuff later! :-P

Examples of paragraph opening/closing tags and html opening/closing tags:

- There are two types of Open/Close tags:
  - Open/Close tags that you write text in. Examples:

     write text - text will be placed on the website..

<h1>write text</h1> - the text will be placed on the website and automatically have a bigger font-size.
<title>write text</title> - the text will appear in the tab bar on the browser.
<a>write text- text will be used as a link.
<button>write text</button>- used to create a button. The text you write will be the name on the button.
 write stuff - used to create a list item.

- Open/Close tags that you put other tags in. Examples:

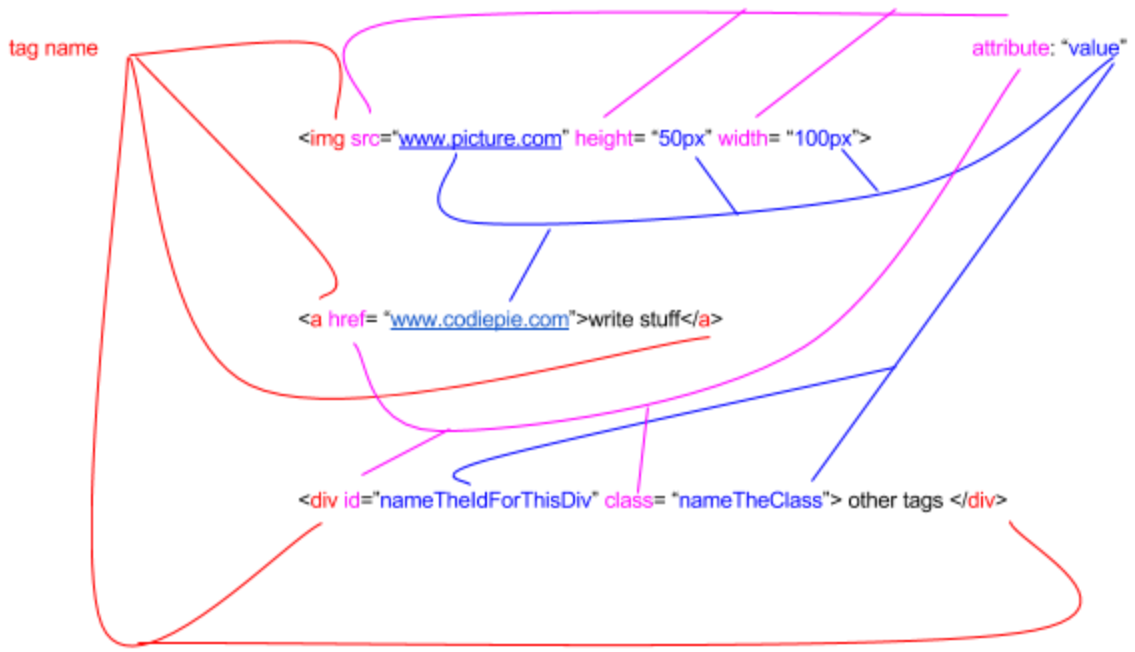
<html> other tags </html> - used to house all of the tags except for the <!DOCTYPE HTML> tag.
<head> other tags </head> - used to house tags that won't be seen on the webpage.
<body> other tags </body> - used to house tags that will be visible on the web page.
 other tags - used to group tags and keep them together.
 other tags - used to group list items together-

- Attributes: Characteristics you can assign to a tag. Some tags REQUIRE attributes or they won't do anything at all. Attributes will have "values". Attributes will be written right after the name of the tag in the opening tag. They will be followed immediately by an equal sign and then its assigned value written inside of quotation marks. Each tag will have its own set of attributes and there can be more than one:

  Basic syntax is:

  <TagName attribute="value">write stuff or other tags</TagName>

  Here are some examples. Notice that there can be more than one set of attributes and values per tag:



- Very special Attributes:
  - style- the style attribute is special because it's values will be a list of properties and values. There can be so many of them, that they can be moved to the <head> section and written between <style></style>
  - ID- an ID is a way of marking an element so that it can be easily identified and targeted for use by the computer. You can make up your own name for the ID.

    Syntax:

    <TagName id="MadeUpIdName"> </TagName>

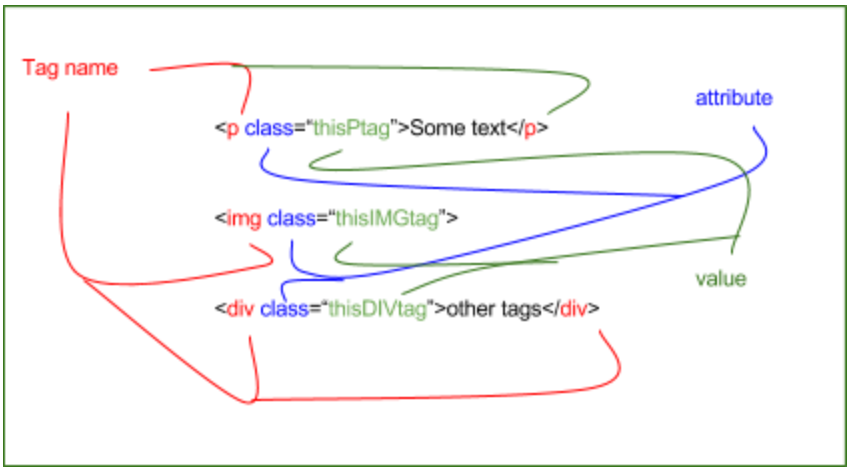
    Examples:

- Class- a way of identifying a group of elements that will be assigned specified style attribute properties and values. Like the ID, you can make up your own name for the class.

  Syntax:

  <TagName class="MadeUpIdName"> </TagName>

  Examples:



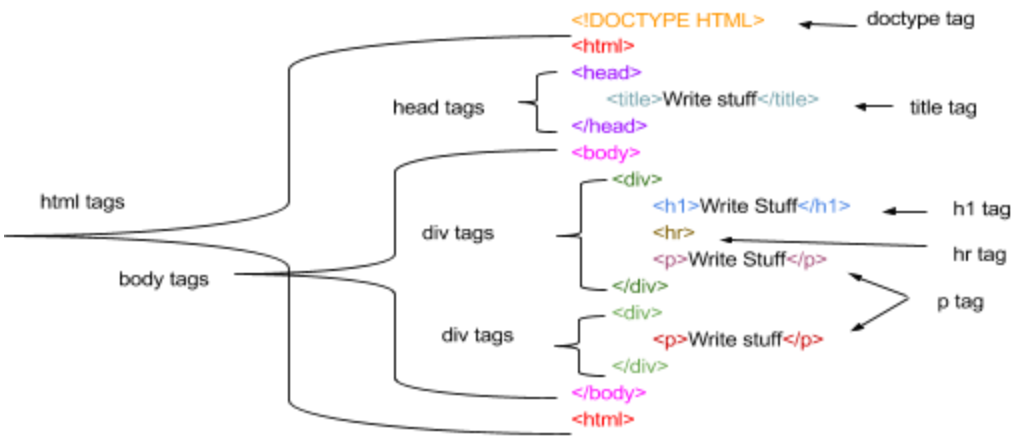
The following p tag has a “style”, id, and style attributes. Notice how it’s value is a list of properties and values. The style attribute is very special and we will examine it in more detail in the CSS section.

```
<p id= “MadeUpIdName” class= “MadeUpClassName” style= “color: red; background-color: yellow; font-size: 20px;”>write stuff </p>
```

- As mentioned earlier, you can place tags inside of tags. Example:

```
<!DOCTYPE HTML> <html> <head> <title>Write stuff</title> </head> <body> <div> <h1>Write Stuff</h1> <hr> <p>Write Stuff</p> </div> <div> <p>Write stuff</p> </div> <body> </html>
```

The above can also be written to make it more understandable. It will look something like this:



Notice how all of the tags that you can write stuff in tend to be written inline while the tags that you put other tags in are spaced in such a way to visualize how they contain other tags and are matched vertically with their closing tags.

- **Assignments/Activities:**

- Explore https://www.w3schools.com/tags/ and look at all of the different tags available to you. Create an html document that makes use of these tags and their different attributes/values. Note: Some tags do absolutely nothing without certain attributes! Here are a list of tags to be sure to experiment with:
    - <img> - **im**a**g**e tag
    - <a> - **a**nchor tag
    - - **o**rdered **l**ist and **l**ist **i**tem tags
    - <button></button> - **button** tag
    - <hr>- **h**orizontal **r**ule tag
    - - **p**aragraph tag
    - <title></title> - **title** tag
    - <input></input> - **input** tag
    - I just realized: the reason why the button and title tags aren't abbreviated is because then they would be called the but tag and tit tag… lol
- Give the student a page of code to "decipher".

- Section 3 - CSS Basics/Syntax
    - CSS stands for Cascading Style Sheets. Style is an attribute that you can find in any tag. Because style values can be numerous, they can also be placed in a separate section between style tags: <style>properties and values</style>.
        - Inline style attribute: Inside a tag, the style attribute can look like this:

```
<TagName style= "property: value;"></TagName>
```

There can me more than one property in the style attribute. Also, some properties can have more than one value:

```
<TagName style= "property: value; property: value value value"></TagName>
```

- Different tags will take different properties. And different properties will have different values.

  Here is a SHORT LIST of examples for just the p tag:

Tag name	property	Possible values
p	border	Pick one of these: None, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, initial, inherit Also can pick a number to indicate pixel size. Otherwise, it'll be 1px. Also pick the color if you want something other than black
	font-size	# of pixels: ie 50px.
	font-family	The name of a font: New Times Roman, Arial, ect

  It would be written as such:

```
<p style= "border: dotted 20px orange; font-size: 50px;  font-family: value:"></p>
```

- Since there can possibly be numerous properties and values, you can also write style properties in a Cascading Style Sheet between style tags which are also between head tags. Here is what it would look like.

```
<head>
<style>
selector {
    property:value;
    property:value value value;
    property: value;
    property: value;
    property: value;
}
</style>
</head>
```

Take note of the selector. As the name implies, it selects what element to assign the style attribute, properties, and values to. A selector can be one of three things:

- A tag name:

```
<head>
<style>
TagName {
    property:value;
    property:value value value;
}
</style>
</head>
```

- An id: to be preceded with a “#” symbol.

```
<head>
<style>
#ID {
    property:value;
    property:value value value;
}
</style>
</head>
```

- A class name: to be preceded with a “.” symbol.

```
<head>
<style>
.ClassName {
    property:value;
    property:value value value;
}
</style>
</head>
```

- **Assignment/Activities:**
  - Explore https://www.w3schools.com/cssref/ properties and values available to you. Create an html document that makes use of and defines a few of these properties.

- Section 4 - JavaScript Basics
    - What is JavaScript.
    - Using the console! We will be using the console to see what the computer is capable of doing using JavaScript!
    - Data types:
        - Numbers- basically numbers like 1, 2, 3, 4,5

          You can do basic operations with numbers in the console:

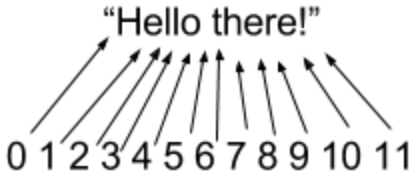
```
3 + 4 → 7
4/2 → 2
8 - 10 → -2
```

5 * 8 → 40

- Strings- text. Strings will be placed between a pair of single or double quotation marks. Examples:

  “Hello World”
  “This is a string”
  “12345”.

  In a string, each character has an Index number according to its location in the order.



  Notice that the first character does not have an index number of 1. Instead, it’s index number is 0. The index count starts at 0 and not 1. Also notice that even the empty space has an index number of 5.

  In the console. You can add strings together:

  “Hello” + “there!” → “Hellothere!”
  “Hello ” + “there with a space included!” → “Hello there with a space included!”

- Booleans- a true or false statement. Can be created using comparison operators:
  - Greater than- “>”
  - Greater than or equal to- “>=”
  - Less than- “<”
  - Less than or equal to- “<=”
  - Equal to- “==”
  - Not Equal to- “!=”

Examples of booleans include:

3 > 2 → true
2 > 3 → false
7 == 4 → false
7 != 4 → true
15 >=15 → true
“String” == “String” → true
“Hello” == “Goodbye” → false
“Hello” == 7 → false
17 <=25 → true

Test them out on the console to see if you get the same results.

Two or more Booleans can be written together to form another Boolean using the following terms:

- And- “&&”
- Or- “||”

They can be used as follows:

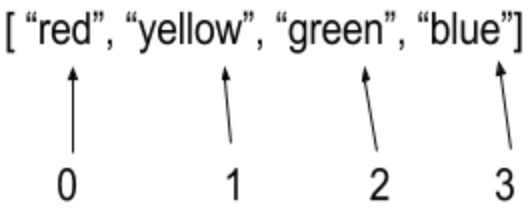
3 < 10 && 15 > 4 → true
3 < 10 && 15 < 4 → false
3 < 10 || 15 > 4 → true
3 > 10 || 15 < 4 → false
3 > 10 && 15 < 4 && 10 > 4 → false
“Hello” == “Hello” && “Hello” != “Goodbye”→ true

- Array- an Array is an ordered list of stuff like numbers, strings, booleans, other arrays, and objects (we’ll be covering objects next). Arrays are signified with “[ ]” symbols with “,” between each thing (aka element) in the array.

  Examples:
  [1, 3, 4, 5, 16, 1, 33]
  [“hello”, “good-bye”, “see you later”, “aloha”]

```
[ 3 < 4, 7 > 3, 7 < 3, 7 == 4]
[ [ 1, 2, 3], [ 9, 10, 15], [ 7, 3, 10] ]
[[ “red”, “yellow”], [ “pink”, “green”]]
```

In arrays, the order matters. Each element has an “index number” according to their spot in the array. They are numbered as follow:

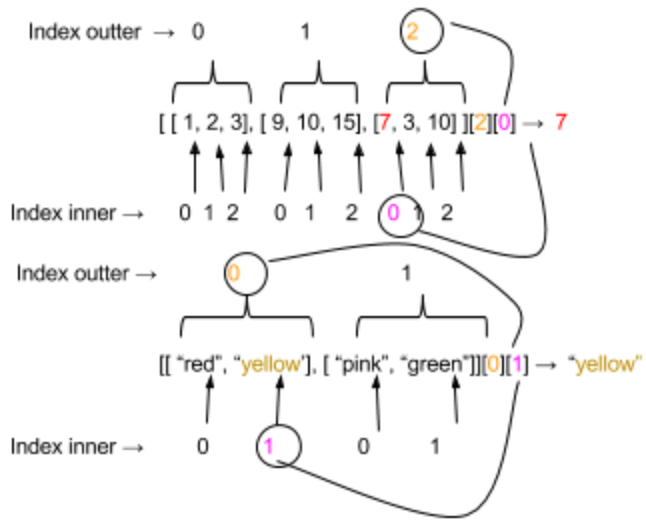


Just like with strings, the first element is 0 and not 1. It’ll be confusing at first but you’ll eventually get used to it.

You can access any specific element in the array by writing it’s index number in a pair of “[ ]”.

```
[ 1, 3, 4, 5, 16, 1, 33][0] --- 1
[ “hello”, “good-bye”, “see you later”, “aloha”][3]--- “aloha”
[ 3 < 4, 7 > 3, 7 < 3, 7 == 4][2] --- false
[ [ 1, 2, 3], [ 9, 10, 15], [ 7, 3, 10] ][2] --- [ 7, 3, 10]
[ [ “red”, “yellow”], [ “pink”, “green”] ][ 0 ] -- [“red”, “yellow”]
```

To access an element inside of array that’s inside of an array(aka a “nested array”), you’ll need two sets of brackets. One that give the index of the outer array and a second to give the index of the inner (nested) array:



- Syntax
- Camel-case for variables, id, classes, and function names
- Functions, “name space”, arguments, “use strict”, string, “return”
- The Stack- How the computer carries out functions.

```
<script>
function alertMessage(message){ // Camel-case your functions
        'use strict'; // Strict mode to prevent runtime error lag
            alert(message); // Initiate the pop-up and print the message variable
    }
    alertMessage('Hello'); // This is a shortcode to activate the alert message
</script>
```

- Section 5 - Built In JavaScript Functions
  - What are they?
  - Commonly used functions:
  - **Resources**
    - https://www.tutorialspoint.com/javascript/javascript_builtin_functions.htm
- Section 6 - Inspirational Javascript Projects
  - Looking into ThreeJS
    - Library functions

        - Projects
- Section 7 - Grid Layouts/Columns/Templates
    - Full, half columns
    - Visual coders- Mantis
- Section 8 - Forms and Input
    - Understanding the syntax
        - Shift and dropping of the “/” from “/>”
    - Form Attributes
        - Action
        - Methods
    - Input Attributes
        - Name
        - Maxlength
        - Placeholder