Code Smarter

With

Code JS

Framework

Table of Contents

INTRODUCTION TO CODE JS

History

In the year 2022, I upgraded my ability to solve web frontend problems and even wrote my first web game (https://spellingbee1.vercel .app). Then I made few other games, because making websites was a bit boring. Then making more apps and games no longer became a major objective. I was still unsatisfied and wanted to do more than just writing apps and games. I thought of what I could create as a programmer that would help/benefit other programmers, then I thought of creating a very little library with some code snippets. I started out crazily and the end result was. After a few months, I decided to remake it and make it more complex, consisting of functions and some extra features that lets you bind and call application data and even external variables whenever and wherever you need them. I also added some special code to help solve some special problems, especially in the area of design and functionality. The former (nano-sized) library, now a (micro) framework is finally ready.

What is Code JS?

Code JS is a JavaScript Frontend Development Framework that focuses on HTML DOM and lets one manipulate it easier and faster. With Code JS, you can;

- Build & manipulate web contents.
- Bind application and Meta data.

- Wrap new styles for objects in a single JS function.
- Directly insert variables from an external JS file.
- Import your stylesheets and scripts without directly insert them in your index file.
- Create single page applications.
- Manipulate contents faster with its functions.
- Build your web apps & contents with JS easier, if you do not wish to write HTML directly in your markups & lots more...

Requirements

To use Code JS, you must have a basic knowledge and understanding in;

- **HTML5** - Standard Markup Language for structuring, and defining the layout of a webpage/web app.

- **CSS 3** – Used for styling a webpage/web app.

- **JavaScript** - A programming language, basically for programming the behaviour of a webpage/web app.

More Info on Code JS

Code JS is a framework that tests & proves how smart & capable people are in solving web frontend problems in design and functionality, while providing them with tools and functions to help them code faster on a large scale.

WORKING WITH CODE JS TEMPLATE

THE AIR

The AIR is simply an acronym for:

app.js – This is the main script which will hold every JavaScript code for developing the web app, and will also be used to import/add other external JS scripts to the page. It can be done using any of the import functions in Code JS framework and will be talked about later. This file is and must always be in the JS folder.

Index.html – This is the main index/homepage file.

root.css – This is the main file that will contain all our CSS code which we'll use to add/import other external stylesheets to our page. This importation can simply be done using our CSS @import rule. This file is and must always be in the CSS folder.

Note: Others are folders that hold files, pages etc.

The I
(index.html)

UNDERSTANDING DATA INTERCHANGE & BINDING WITH CODE JS FRAME

Code JS lets you bind your major web app data using the <**application /**> element. This element binds our application data and contain the three main elements for our app development.

The following are attributes of the **<application />** element:

name – This holds the name of the application/website and sends it to the head element under the title element and application-name attribute in the meta tag/element.

logo – This stores your application/website logo and places it where it's needed. As we continue the journey, you'll understand fully.

author – This stores the name of the author/company that owns the app.

theme – This is used to specify a theme (major colour) for your application.

mode – This is used to specify a theme between (light or dark) for the entire app.

Sample:

```
<application name="" author="" logo="" theme="" mode=""></application>
```

There are three sections in the application element that contains our entire (visible) content. We have the;

<data></data> section -- For containing our meta data.

<layout></layout> section -- For containing our HTML code.

<engine></engine> section -- For containing our external scripts (codejs.cjs & app.js).

As you can see, all meta tags are contained with the **<data>** element, and are binded in three lines. One for the **<meta charset>**, then the **<meta-name>** (instead of **<meta name="" content="">**, and the **<meta-http-equiv>** (instead of **<meta http-equiv>**), as shown below;

```
<data>
    <meta charset="charset">
    <meta-name name="content">
    <meta-http-equiv name="content">
</data>
```

The **<layout />** element contains our markup as shown below;

```
<layout>
    <header></header>
    <nav></nav>
    <main>
        <section>
            <article></article>
        </section>
    </main>
    <aside></aside>
    <footer></footer>
        </_Comments goes here...>
```

And comments in HTML can also be written as;

```
        </_Comments goes here...>
```

Then we have the <engine></engine> element for containing our scripts, including adding from other sources.

Note:

All existing files in the template have been completely linked/wired together, so do not link again to avoid unnecessary errors. Goto css/stylesheets.css to import css stylesheets or libraries, even without adding <link /> to the index.html, but in case, you can use the <link /> in <data> element, after the <meta-http-equiv /> element. Use imp() function in your app.js file to import another external script, or importAll to import an array of scripts) maybe with conditionals(if...else statements) or within functions, and use impX() function to import scripts from different servers, you'll see examples later.

You can insert JavaScript variable/constant data from an internal or external source using "compile" attribute, and <js> element. For example;

Create a new variable called data or greetings in app.js (or any external script)...

app.js:

```
var data = "Hello World";
var greetings = “Welcome to my world!”;
```

index.html

```
<layout>
        <button compile> data </button>
         <h1>
                <js> data </js>
        </h1>
        </_Using `backtick` and curly braces>
        <js>`
                <h2> ${data} </h2>
                {{greetings}}
        `</js>
        </_Inserting math calculations>
        <js> (2+2) * 4 – Math.sin(2) / 80 * 200 </js>
        <js>` I am 50 * 50 years old `</js>
</layout>
```

Then Check the result! You can check your console to confirm the new comment tag (</_Comment>).

You can also run and display math calculations either using the compile attribute or the JS element.

The A
(app.js)

Framework/Library Data

To see the version of Code JS that you're currently using, write ***CodeJS.version*** and you'll get it.

The App Object

app.name() – This returns the application's name.

app.author() – This returns the application's author.

app.logo() – This returns the application's logo.

app.theme() – This returns the current theme of the app.

app.mode() – This returns the current mode of the app.

app.copyright() – This returns the copyright (& year of development) of the app.

app.backgroundData() & app.messages() – These, when used at once, returns/tells a user how long he has spent on an app. To modify it, use the browser console to get the attributes (ID/Class), then manipulate.

app.footnote() – This function generates a footnote to your web app.

Selectors

The following are used to select HTML elements in Code JS;

$(element_name) – This is used to select single elements.

$$(element_name) - This is used to select a group of elements.

Note: Both work just like CSS selectors, that is they let you use # for ID, "." for class (class names) and "[ ]" for attributes and attribute values. The group selector works with Code JS loop functions, which make iteration for both arrays, objects and HTML elements easier and faster.

ROM Components

Code JS ROM Components are components containing special data for special purposes and which are called upon wherever necessary. This components have already been created and stored in Code JS and their sole aims/purposes are for the advancement and development of websites / web apps.

There are two (2) types of ROM Components in Code JS.

i) **Single ROM Components:** These are components that were built with only JavaScript.
E.G: Date components...

ii) **Combined ROM Components:** These are components that were built by combining JavaScript and other languages like HTML5 & CSS3.
E.G: Code JS Objects (*Will be introduced later*)...

Code JS Date

This object is used to insert date to your web page / web app. In your layout element in the index.html file, write:

<js> d.fullDate() </js>

And preview to see the full date according to device's system's time.

You can also write to your app.js file:

html($('layout'), d.fullDate());

It'll still give the same results.

Here's a full list of all the date functions:

d.day()

d.date()

d.month()

d.year()

d.fullDate()

Code JS Constructors

If you want to build/design and program a web app with JavaScript, Code JS Builder (building functions) will help you accomplish it easier and faster.

Layout Constructor

If your website contains one header, navigation (nav), main (for the major contents including sections and articles), aside & footer, simply use the **layout.create()** function to create all at once, e.g:

layout.create();

And, use **layout.destroy()** function to complete empty/destroy the layout (contents.

Note: You can check your console to see messages.

Widgets Construction & Rendering

In Code JS, HTML elements are treated as widgets and are given the default class "widget" when created.

To create an element, use the widgets.construct() function.

Syntax: widgets.construct(element)

To add the element to your layout, use the render() function.

Syntax: render(element, target) => render(widget, $('layout'));

To create multiple elements at once, use widget.constructMultiple() and use renderAll() to generate all at once.

Note: You can also use any of the render() functions to move an already existing element from one place to another, on the web page.

Other Code JS Functions

1. **renderBefore(widget, target) –** renders elements before the target on the web page.

2. **renderAfter(widget, target) –** renders elements after the target.

3. **renderFirstPlace(widget, target) –** renders elements as the first child of the target.

4. **renderLastPlace(widget, target) –** renders elements as the last child of the target.

5. **pushBefore(target, "contents") –** pushes HTML contents as the first content of the target.

6. **pushAfter(target, "contents") -** pushes HTML contents as the last content of the target.

7. **removeContent(target, "contents") –** searches and removes contents from the target.

8. **replaceContent(target, "existing contents", "new contents") –** searches and replaces contents in the target with new contents.

9. **removeContents (target, “contents”) –** searches and removes all same contents at the same time.

10. **replaceContents(target, “existing contents”, “new contents”) –** searches and replaces all same contents at once with a new one.

11. **insertBefore(target, “str”, “new content”) –** searches and adds new contents before a string on the web page.

12. **insertAfter(target, “str”, “new content”)** – searches and adds new contents before a string on the web page.

13. **insertBeforeAll(target, “str”, “new content”) –** searches and adds new contents before all same string on the webpage.

14. **insertAfterAll(target, “str”, “new content”) –** searches and adds new contents before all same string on the webpage.

Widget Cloning

15. **clone(target, widget/selected elements) –** used to clone a widget into a target.

16. **cloneRepeat(target, widget) –** This lets you clone the same widget into a group of targets.

17. **clones(target, obj, n) –** This lets you drop multiple clones of a single widget into a single target.

18. **swap(widget 1, widget 2) –** This lets you swap widgets.

Content Manipulation

Contenting

19. **html(target, content) –** Lets you return HTML contents of a widget (using **html(widget)**) and even lets you rewrite them (using **html(widget, new contents)**).

20. **text(target, content) –** Lets you return text contents of a widget (using **text(widget)**) and even lets you rewrite them (using **text(widget, new contents)**).

21. **attrib(target, attribute, data) –** Lets you return the attribute data/value of a widget (using **attrib(target, attribute)**) and even lets you rewrite them (using **attrib(target, attribute, data)**).

Styling

22. **css(target, "attribut1 e: value 1; attribute 2: value 2;...") –** Lets you define a series of new styles for a widget.

Functionality

23. **tplclick(target, func) –** This lets you create triple click functionality just like doubleclick in HTML5, then place your statements, just like this **tplclick(target, function () => {statements}.**

24. **restrict(target) –** This function disables a widget.

25. **restrictAll(target) –** This function disables an entire group of widgets.

26. **unrestrict(target)** – This function enables widgets.

27. **unrestrictAll(target) –** This function enables an entire group of widgets.

Animation

28. **animate(target_container, time) –** This function places widgets within a container or code block in animation mode.

Importing Scripts

29. imp(src) – This function, which is already connected to the JS folder lets you import scripts within the JS folder.

30. **impX() –** This function lets you import scripts from other sources/servers.

31. **importAll(array_of_scripts/sources) –** This function lets you import an array of scripts, from the JS folder.

Visibility State

32. **hide(widget) –** Lets you hide a widget.

33. **hideAll(widgets) –** Lets you hide a group of widgets.

34. **unhide(widgets) –** Lets you unhide a widget.

35. **unhideAll(widgets) –** Lets you unhide a group of widgets.

36. **del(widget) –** Lets you delete a widget.

37. **delAll(widget) –** Lets you delete a group of widgets.

Pick a Random

38. **randomize(widget) –** Lets you display random data or widgets from a group of data or widgets.

Iteration

39. **randomSelect(group, target_value, func) –** This function lets you pick a limited random group(target value) from another group.

40. **loop(group, func) –** This function, just like the for loop lets you iterate groups.

41. **loopX(group, func) –** With this function, "I" becomes "group[i]" and is used instead of "group[i]".

42. **rangeSelect(start, end, func) –** This lets you select/iterate fo a group within a specific range.

43. **singleSelect(group/array, func) –** This lets you manually iterate and select from a group.

44. **typeSelect(group, type, func) –** This lets you iterate a group by "even" or "odd" (numerically).

Integers

45. **randomOf(num, func) –** This returns a random integer/number from num(length).

46. **arrayOf(num, func) –** This returns an array of number (1 – num).

Tracing & Untracing Elements

47. **trace(widget) –** Lets you trace the child elements of a widget.

48. **untrace(widget) -** Lets you untrace the child elements of a widget.

Switch between Contents

49. **switch_content(class, id) –** This, being the only Code JS content manipulation function that does not require you to

select a widget, but lets you switch between contents of a web app, determining which to display per time or event, using their classes and ids.

Popping Widgets

50. **popup(widget) –** Lets you pop a widget to the screen.

51. **popdown(widget) –** Lets you remove the popup.

CSS Transform Functions

The following functions are identical (has same functionalities) to the CSS 3 transform properties.

52. **rotate(widget, value)**

53. **rotateX(widget, value)**

54. **rotateY(widget, value)**

55. **rotateZ(target, value)**

56. **translate(target, valueX, valueY)**

57. **translateX(target, value)**

58. **translateY(target, value)**

59. **scale(target, value)**

60. **scaleX(target, value)**

61. **scaleY(target, value)**

62. **skew(target, value)**

63. **skewX(target, value)**

64. **skewY(target, value)**

Binding data to Iframe

65. **bindToFrame(target, obj) –** This function lets you bind any data to an iframe.

Adding & Removing Classes

66. **removeClass(target, value) –** Lets you remove a class.

67. **addClass(target, value) –** Lets you add a class.

Audio Callbacks (For Chrome and Edge Browsers)

68. **voice.output(data)** – For audio callbacks. Supported majorly by Chrome & Edge browsers.

Events Handling

Code JS lets you add event listeners by calling them as functions and adding parenthesis within them. For example;

Use click(element, () => {

alert('hello world!');

}); instead of element.addEventListener('click', function () {

alert('hello world!');

});

Same thing goes for every other existing events.

Objects

Code JS Objects, also called Combined ROM Components are very interesting predesigned/readymade widgets/objects that are used to support your frontend application development. These include:

69. **loadApp() –** This function is used to create a loader for your app. It binds your app logo and theme color to a container and displays it when a person visits your website/web page/web app.

70. **loadContent(func) –** This function is used to load specific contents. It's was created majorly for loading contents when a link or button is pressed, and you were to call other functions after a link was clicked.

INTRODUCING CODEJSL.CJS

This is Code JS Library that lets you use the Code JS major functions for content manipulation in your web pages other than your index page.

ACHIEVEMENTS

From designing and manipulating the visible aspects of a website/web apps to building games, Code JS has been used to make web coding and web frontend project management easier and faster.

THE DEVELOPER

Ronaldo Excellent

Social Contacts:

Tel: +234 9114349749
Facebook – https://facebook.com/ronaldinhoexcellent
Messenger – https://m.me/ronaldinhoexcellent
Instagram – @ronaldo.excellent
Twitter - @ronaldoexcel
Github – https://github.com/ronaldoexcellent
Reddit – https://reddit.com/user/ronaldoexcellent
Buy Me A Coffee – https://buymeacoffee.com/ronaldoexcel