Bracket: Web Development Technology

Bracket is a live web development technology and a programing language that interacts between html, js, and css, and renders on both server and browser sides, where a developer writes only Bracket code to build a web application. In every web application there are views styled with controls. The philosophy of development in Bracket is that you choose first the view you want (text, image, icon, input, view...), then you style it, add some parameters, and finally add controls (events, functions, methods) to interact.

Why Bracket

What makes Bracket language mostly different and better than any other language is how much easy it is to learn and write. Its name Bracket is derived from the syntax of it. The round brackets "()" alone call a view and the parameters inside it, the opposite brackets ")(" call the global store, the round brackets "()" added to a method name call methods and functions, the square brackets "[]" is used for parameters and conditions (ex: "[input.type=text]"), and used for objects like "string:[color=red;fontSize=1.3rem]".

An example of using different kinds of brackets together "if():[!readonly]:[txt()=Hello World]". "if():[conditions]:[params]" is a function executes if conditions are applied.

There are another reserved punctuations like "?" ":" "." ";" "=", however you could use any of them as a text string by putting the text between 2 quotes

(ex1: text='First Name is: Mohamad' | ex2: text='What are your hobbies?').

Start coding with Bracket

To start building an application using bracket, follow these steps:

1 .Signup on bracketjs.com

2 .Create a project

3 .Start creating pages and views

When a developer arrives the developer editor page in bracket, he is ready now to start coding. By default a page (named "main") and a "main" view are created. The main page is the landing page of the developer web application with path "/", and the "main" view is a view included in the "main" page.

Structure

The structure of bracket code is JSON. Every file (view, page, ...) must have specific fields and values that will be converted to html, css, and js and will interact according to bracket logic.

View

A View must have a type (Image, Icon, Text, Input, or View), and could have children (another views inside this view), and controls. The value of type is structured as follows: "view_type?parameters?conditions". This means that a view is conditional and has params (like styles, events, id, class, parameters related to the view type...) that will be assigned to the view or executed on a specific event. So a field value has the value itself, params, and conditions that are separated by question marks. Ex. of a text view:

{ type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!" }.

Here there are no conditions to be considered, the view type is Text, and the text has some styles and its content is "Hello World!". Parameters (also conditions) are separated by semicolons as shown in the example above.

A view is an object with many parameters that has a parent view and may have children views. A child could be also a parent of other children views. Any view could be included in any and many pages.

Page

Every page has a unique path, unique id, title, controls, any params, and views (multiple views). Ex: The home page has a navbar, sidebar, body, footer... each one is considered as a view that could be created separately (in a separated file) or included all in a one file.

Stores

In Bracket there are two places to store values. The first one is global and is symbolized as ")(" and is accessible easily anywhere in the app but does not affect the views. The other store is each view itself and is symbolized as "()" and also accessible all over the app. To access a view, there are several ways to do that. By using its id, or calling its parent, or calling its sibling, or calling a child of its children... but the most popular way is using id (ex. "():id"). Adding no id (just using round brackets "()") means you are accessing the local view.

Controls

There are different ways to add controls. Ex you want to change the text color in the previous example color when mouse hovers over it to blue. The first way to do that is adding another parameter to the view type field.

Way 1:

{ type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!;hover.style.color=blue" }

Way 2:

{ type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!;mouseenter.style.color=blue; mouseleave.style.color=red " }

Way 3:

```
{
type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!,
controls: [{
        event: "mouseenter?style().color=blue"
}, {
        event: "mouseleave?style().color=red"
}]
}
```

Note: 1. style() is a method to change the view style.

To change more than one style at the same time (add a background color) you could event add them each as a separated parameter or write as follows.

Way 1:

{ type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!;hover.style:[color=blue;backgroundColor=yellow]" }

Way 2:

{ type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!;mouseenter.style:[color=blue;backgroundColor=yellow];
mouseleave.style:[color=red;backgroundColor=inherit] " }

Way 3:

{

type: "Text?style:[color=red;fontSize=15px;backgroundColor=yellow];text=Hello World!,

controls: [{

event: "mouseenter?style():[color=blue;backgroundColor=yellow]"

}, {

event: "mouseleave?style():[color=red;backgroundColor=yellow]"

}]

}

Methods and Functions

There are many implemented functions and methods in Bracket that makes coding faster and easier, and also many functions are at the same time methods that could be used as you wish.

Examples of methods and functions: split(), slice(), splice(), find(), _find(), filter(), _filter(), (), _(), sort(), search(), save(), erase(), contains(), is(), isnot(), inc(), rem(), del(), flat(), children(), 1stChild(), 2ndChild(), parent(), lastChild(), 2ndLastChild(), siblings(), deep(), import(), export(), replace(), timestamp(), getDateTime(), timer(), interval(), date(), toSimplifiedDate(), toClock(), today(), keys(), values(), txt(), pull(), pullItem(), push(), max(), min(), clone(), src(), sum(), opp(), data(), in(), pushStat(),

replaceState(), clearTimer(), mobile(), desktop(), tablet(), mouseenter(), mouseleave(), click(), focus(), className(), position(), getInput(), getTag(), style(), prev(), next(), 2nd(), 1st(), last(), if(), while(), setCookie(), getCookie(), eraseCookie(), doc(), win(), e(), type()... and many many other implemented methods/functions.

Development

Development environment is live on internet. When editing your code and after saving, the changes could be watched immediately on the app, ex: on my_project_name.bracketjs.com/page_name.

A developer using bracket could create many collections of products, hotels, reviews, or any collection type, and could store files and images.

Host & Data store

Data of each project (the views, pages, collections, products, and any other data) are stored in firestore, and project is hosted on firebase (for now and soon on AWS EC2) as well.

Build now

Bracket language compilation is very high in speed, efficient, and very stable. It is still under development, and the platform under construction, even though a developer starting from now is able to build lots of applications using Bracket platform.

Applications Under Construction:

1. https://alsabil-tourism.bracketjs.com (A flight reservation system, currently only flights can be created, listed, and searched for using search engine)
2. https://bracketjs.com (The platform for building Bracket web applications)
3. https://beirut-group.bracketjs.com (A hotel reservation system, currently under construction)
4. https://snack-mart.bracketjs.com (An online shopping market for snacks and drinks, currently under construction)

BIIS Account to code with Bracket: https://bracketjs.com/developer-editor/biis

Full Name: Mohamad Baker Obeid

Location: Lebanon, Beirut

Email: komikbenim@gmail.com

Phone-number: 81-026725