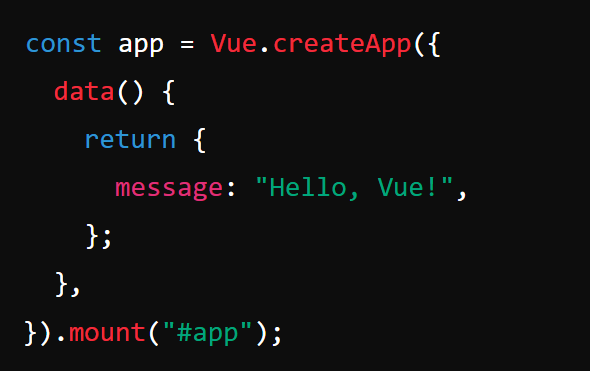
1-Data Binding:

Data binding in Vue.js allows you to bind DOM elements to your Vue instance's data properties.

This means that any changes in the data are automatically reflected in the DOM and vice versa.

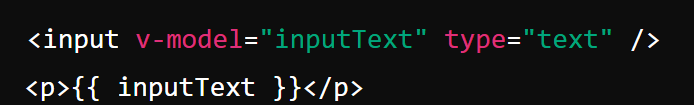
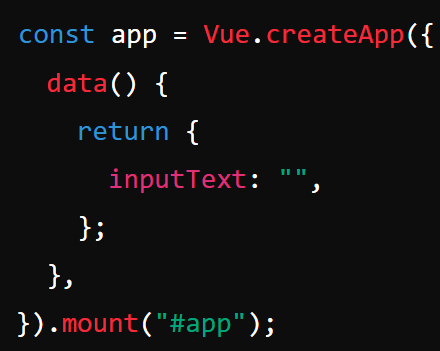
Types of Data Binding:

Interpolation ({{ }}): The simplest form of data binding, where you display the value of a data property in the DOM



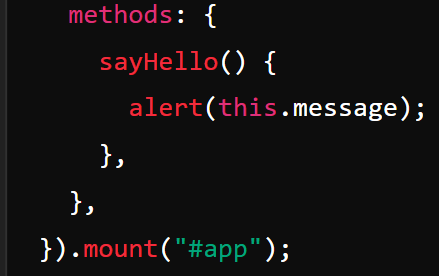
**Attribute Binding (v-bind: or :)**: You can bind data to HTML attributes like class, style, src, etc.

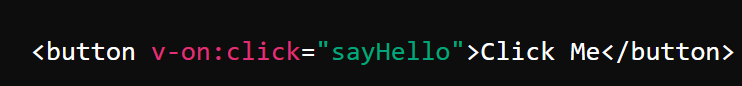
**Two-Way Data Binding (v-model)**: This is a special directive that allows you to create a two-way binding between form input elements and data properties. This means that changes in the input element will update the data property, and vice versa.

يعني بدل ما هتعمل ليسن علي الإنبت وبعدين تروح تحط الفاليوا بتاعته فيه برده عشان لو غيرت فيها تتشال من الإنبت بعدل ما هتحط اتنين هتستعمل ال v-model => v-bind:value + v-on:input=” اسم الفنشكن ”

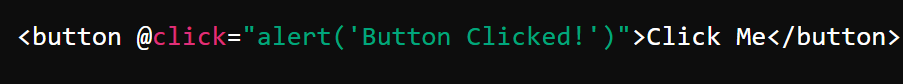
Here, the inputText property is bound to the input field. Any change in the input field will update the inputText data property, and any change in inputText will be reflected in the input field.

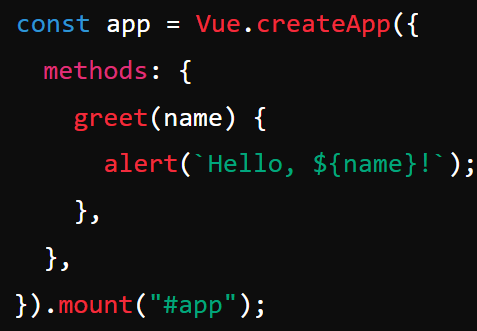
2. Event Binding

Event binding in Vue.js allows you to listen for and respond to DOM events (like clicks, keypresses, etc.). Vue provides the v-on directive (or the shorthand @) for this purpose.

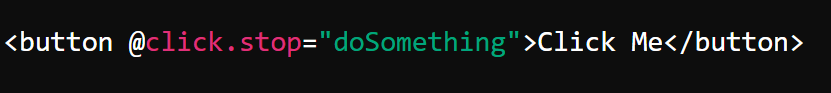
**Basic Event Binding**:

**Event Binding with Inline Handlers**:



**Passing Arguments to Event Handlers:**

Vue.js provides **event modifiers** to handle common event behavior easily:

* **.stop**: Stops the event from propagating.
* **.prevent**: Prevents the default action associated with the event.
* **.capture:** Adds the event listener in the capture mode.
* **.self**: Only triggers the event if it was triggered on the element itself, not its children.
* **.once:** The event will be triggered only once.

Summary

* Data Binding: Connects your Vue instance's data with the DOM, allowing automatic updates and synchronization.
* Event Binding: Allows you to listen to and respond to user interactions (events) on the DOM elements.

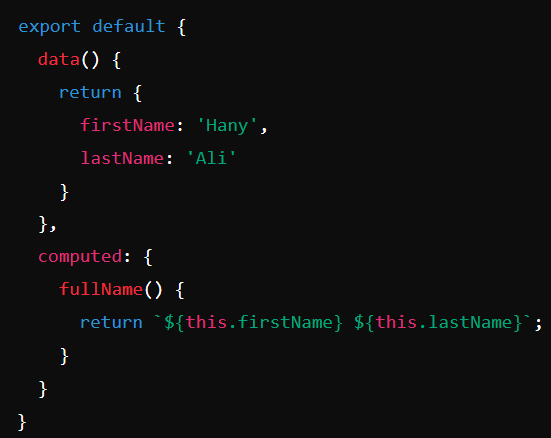
**Compare between watcher, computed and methods in Vue and when we use ease one**

**They are three important concepts used to handle reactivity, data manipulation, and business logic.**

**Computed Properties:**

**Purpose: Computed properties are used to derive new data from existing data. They are cached and will only recompute when their dependencies change.**

**Usage:**

* **When you need to derive data: If you want to create a property that is based on other data properties and want this derived property to be automatically updated when the source data changes.**
* **For caching: Since computed properties are cached, they are efficient when you have expensive calculations that don't need to be recalculated unless their dependencies change.**

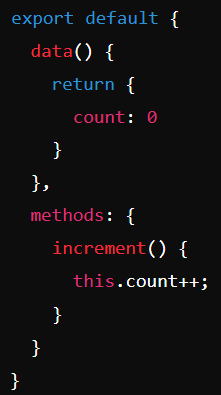
**When to Use:** Use computed properties when you want to create properties based on other data properties, and you want those derived properties to update reactively.

**(بس فيه فرق ان ال الميثد بتقبل بارم ممكن يتبعت داخل الفنكشن انما ال الكمبيوتد مش بتقبل لانها مش فنكشن هيا تعتبر بترجع قيمه زي الفايلوا كدا )**

2. **Methods**

**Purpose:** Methods are functions that are executed in response to events or other actions. They can be used for any kind of logic, including calculations, API calls, and event handling.

**Usage:**

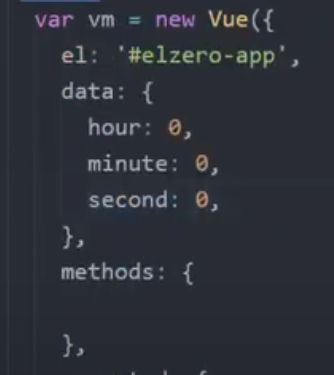
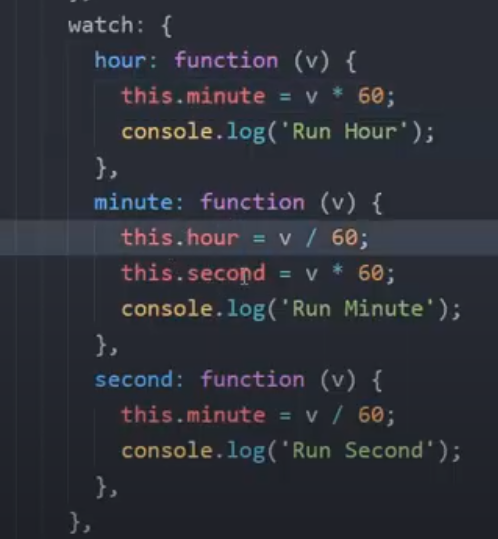
* **When performing actions:** If you want to perform an action in response to a user event, like a button click, or need to execute a piece of logic that does not need to be cached.
* **For dynamic calculations:** When you need to compute something on demand without the need for caching, or if the result depends on parameters passed at runtime.

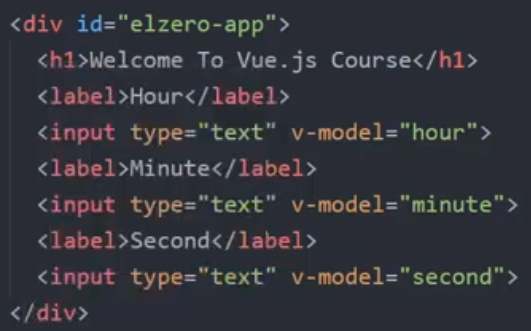
**When to Use:** Use methods for event handling, performing actions, or when you need logic that doesn’t depend on caching or reactivity.

**3. Watchers**

* **Purpose: Watchers allow you to perform side effects or run logic in response to changes in a specific data property.**
* **Usage:**
  + **When reacting to data changes: Use watchers when you need to perform an action whenever a specific piece of data changes. For example, making an API call when a data property changes or validating input in real-time.**
  + **For complex logic: When you have logic that needs to execute in response to a change in a data property that might involve multiple steps, like updating other properties or interacting with external systems.**

**يعني من الأخر عاوز تغير حاجه ال هوا بالمعني الحرفي هتراقبها لو اتغيرت هتغير حاجه تانيه**

****

****

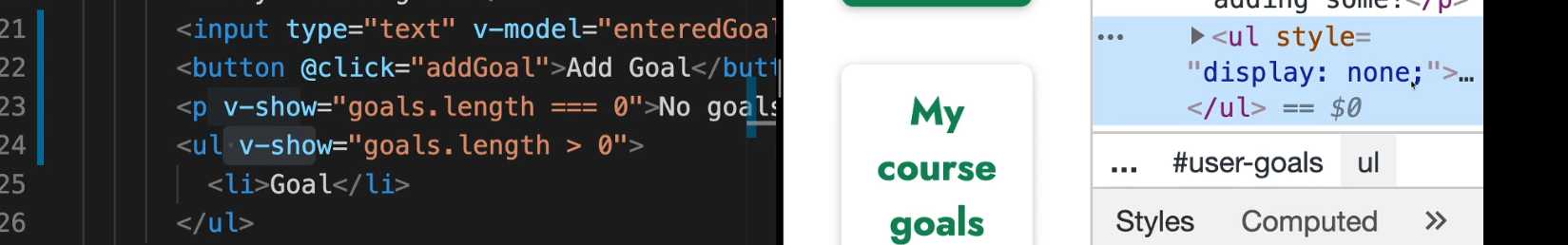
**هنا بقا هتمشي ازاي انت الوقتي عامل اكسس علي الساعات والدقايق الثواني انتا عاوز تقله لما تغير الساعه غير الدقيه والثانيه فهوا عامل واتش ال هوا مراقب ال الساعات اول ما هتغير ف الحقل بتاعه هيعمل واتش علي الدقايق لأكن قيله غيرلي الدقايق بلاقيمه بتاعت الساعه واضربها في 60 فأول حاجه هيطبع ال الكنسل ال في الساعات وبعدين هيروح للدقايق هيا اكدا اتغيرت فبقا فيه قيمه جديده فيها فانت عامل انه هيغير ف الساعات وهيا خلاص اتغيرت فهيلاقي الثواني هتتغير انت كدا عملت واتش علي الثواني فهيطبع الكنسل ال في الدقايق وهيطلع علي الساعات تاني عشان انت استدعيتها جوا الدقايق وبعدين هيروح لل ثواني هيعمل واتش علي الدقايق فهيعمل ال هوا هيطبع الكنسل ال ف الثواني وبس كدا**

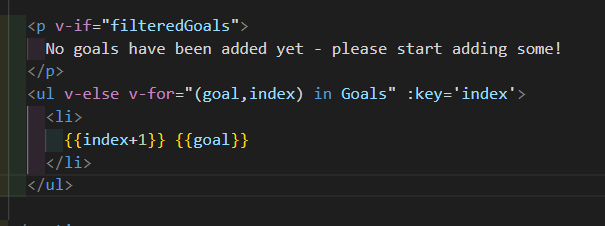
**When to Use:** Use watchers for side effects or when you need to run code in response to data changes, particularly when the logic is too complex for computed properties or when interacting with external systems.

**Summary**

* **Computed Properties: Best for derived, reactive data that benefits from caching.**
* **Methods: Ideal for actions or when you need to run logic in response to events without caching.**
* **Watchers: Useful for executing side effects or running complex logic when a specific data property changes.**

**V-if, v-else, v-else-if, v-show**

**V-if, v-else, v-else-if=>control the element and if the condition is false the element removed from the Dom, but v-show if the condition is false the element just make hidden by display none** ****

****v-for like for loop asdf

Key like one in map

**What is proxy**

**Vue Instance lifecycle**

**1- Start with createApp ({..}): this when use create app**

* 1. **beforeCreate():before app have initialized**
  2. **created(): after initialized**

**1.2.1- compile template when convert all js app data to the screen and lead to beforeMount()**

**1.3- beforeMount()**

**1.4- mounted(): this is when all data show in the screen**

**(mounted lead to { Mounted Vue instance } )**

**Mounted Vue instance lead to dataChanged and Instance Unmounted**

**1-DataChanged**

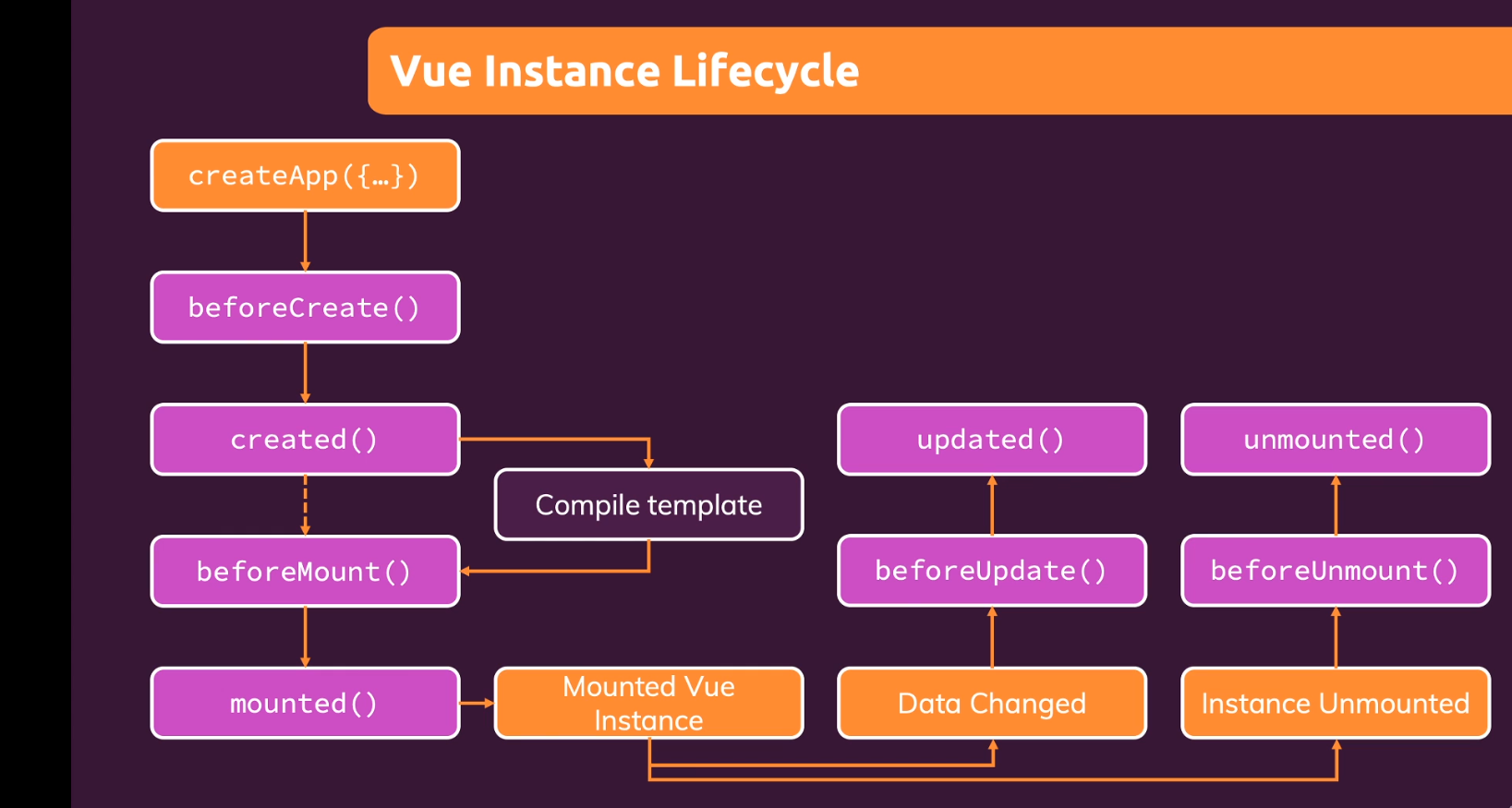
**2.1- berforeUpdate():before data updated when we change it by how**

**2.1 updated():after data change**

**2-instance Unmounted: this will cause dead app**

**2.1- beforeUnmounted(): before kill app**

**2.2- unmounted(): app was killed**

****

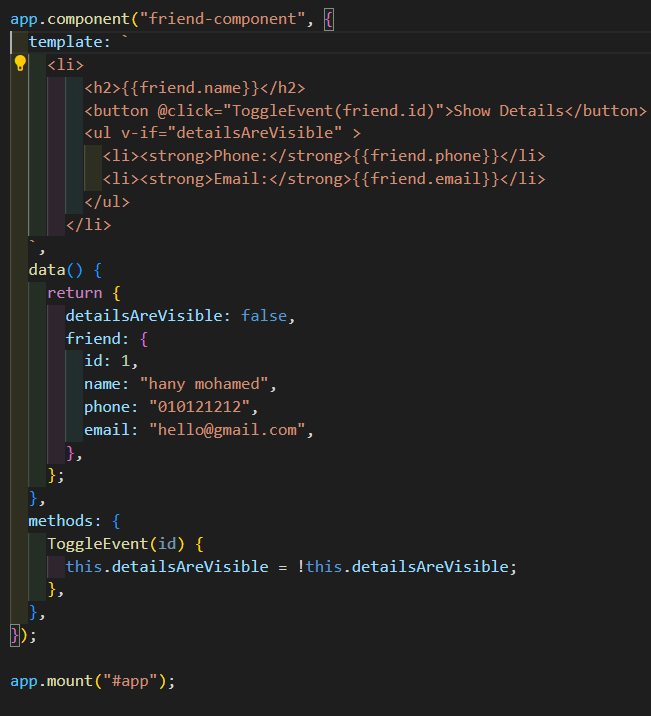
**Start to learn components**

**The first thing you should know you should use dash in the name(-) like =>frind-contact**

**To make sure you doesn’t name you components like html tag like (h2, span, div,….)**

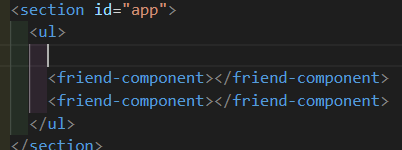
**وخلي بالك ال هوا زيه بالظبط زي ال Vue.createApp**

**يعني فيه كل الخواص ال هيا data,methods,watch,..**

****

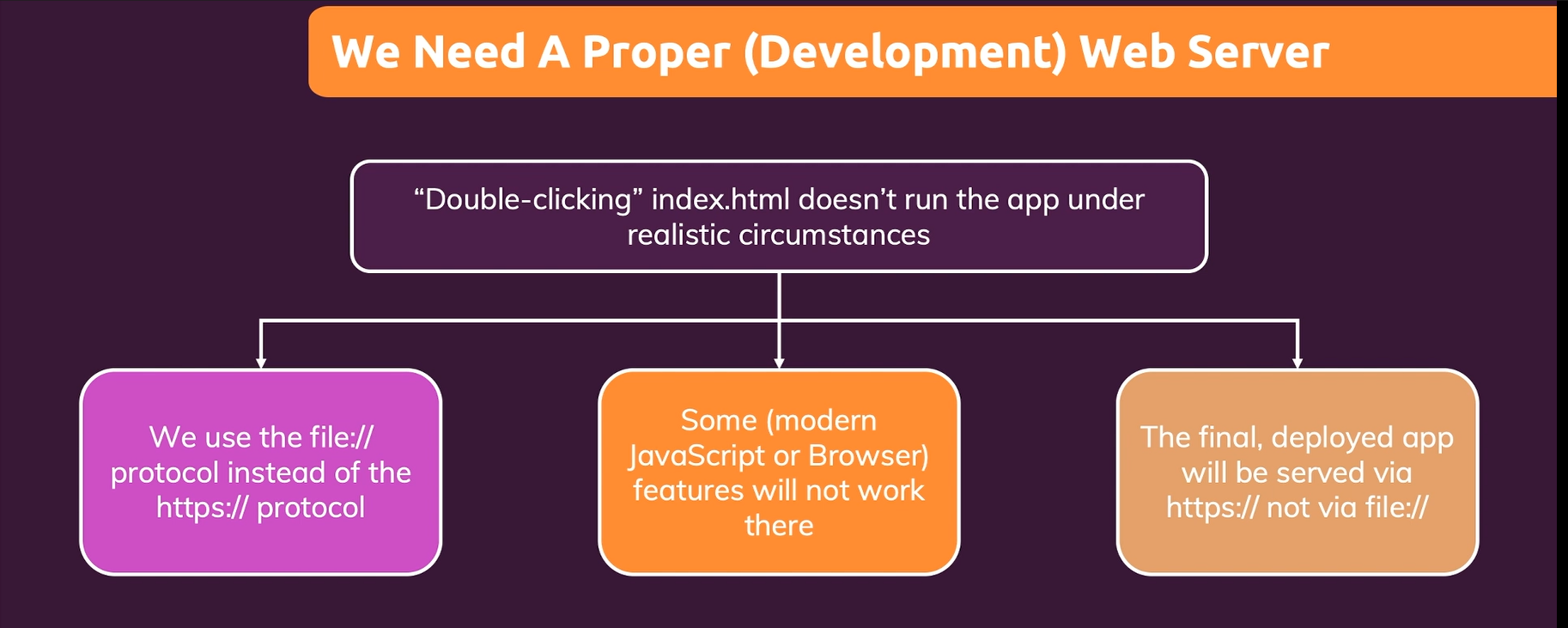
**Template: دا بقا ال هوا المحتوي بدل ما هتكتبه في ال html هتكبه هنا**

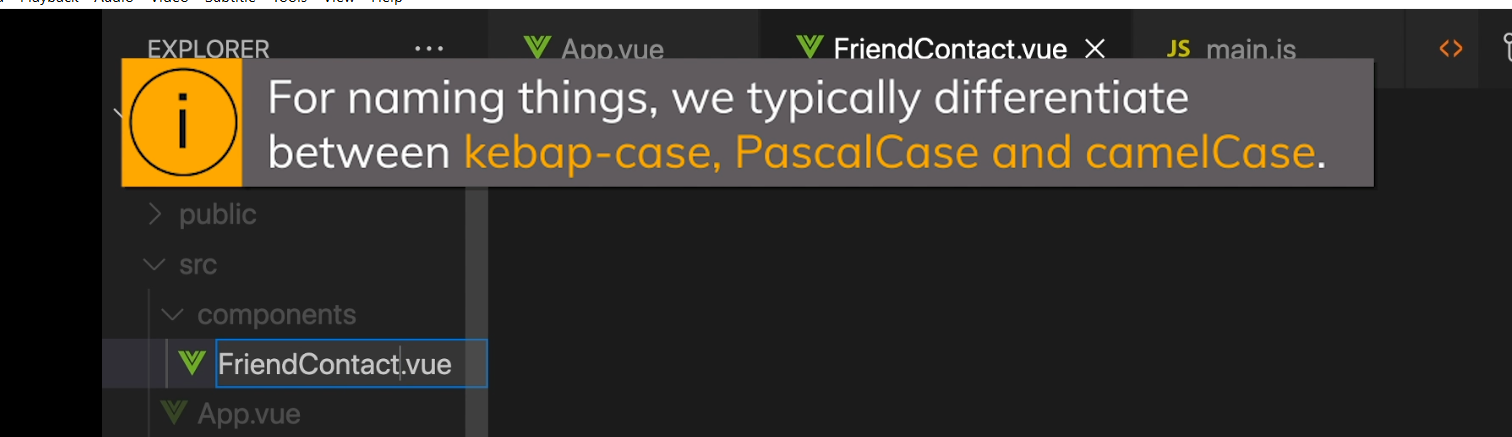
**Data(): دا عادي المحتوي زيه زي ال فوق في Vue.createApp**

** وهكذا في methods**

**ودا بقا لما تيجي تستعمل ال component في الصفحه عشان كدا قلنا نخليه فيه داش عشان ميتعارض مع html element**

**والوقتي خلاص كل عنصر منفرد بذاته عشان لو غيرت ف واحد التاني ملوش علاقه بيه**

****

****

**When you pass props to component pass it as Kebap-case but when you get it talk it as camelCase**



**Like phone-number will be like phoneNumber**