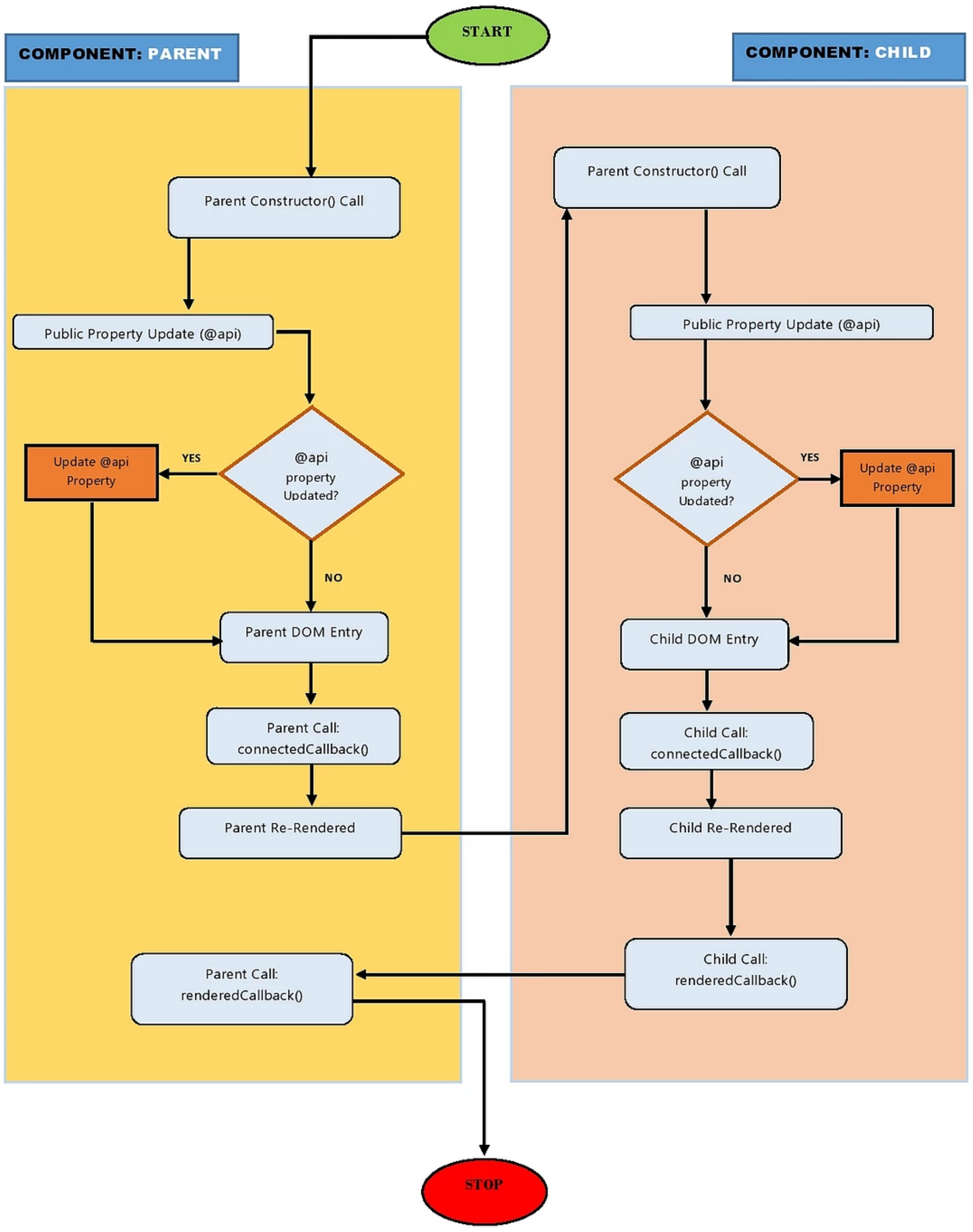
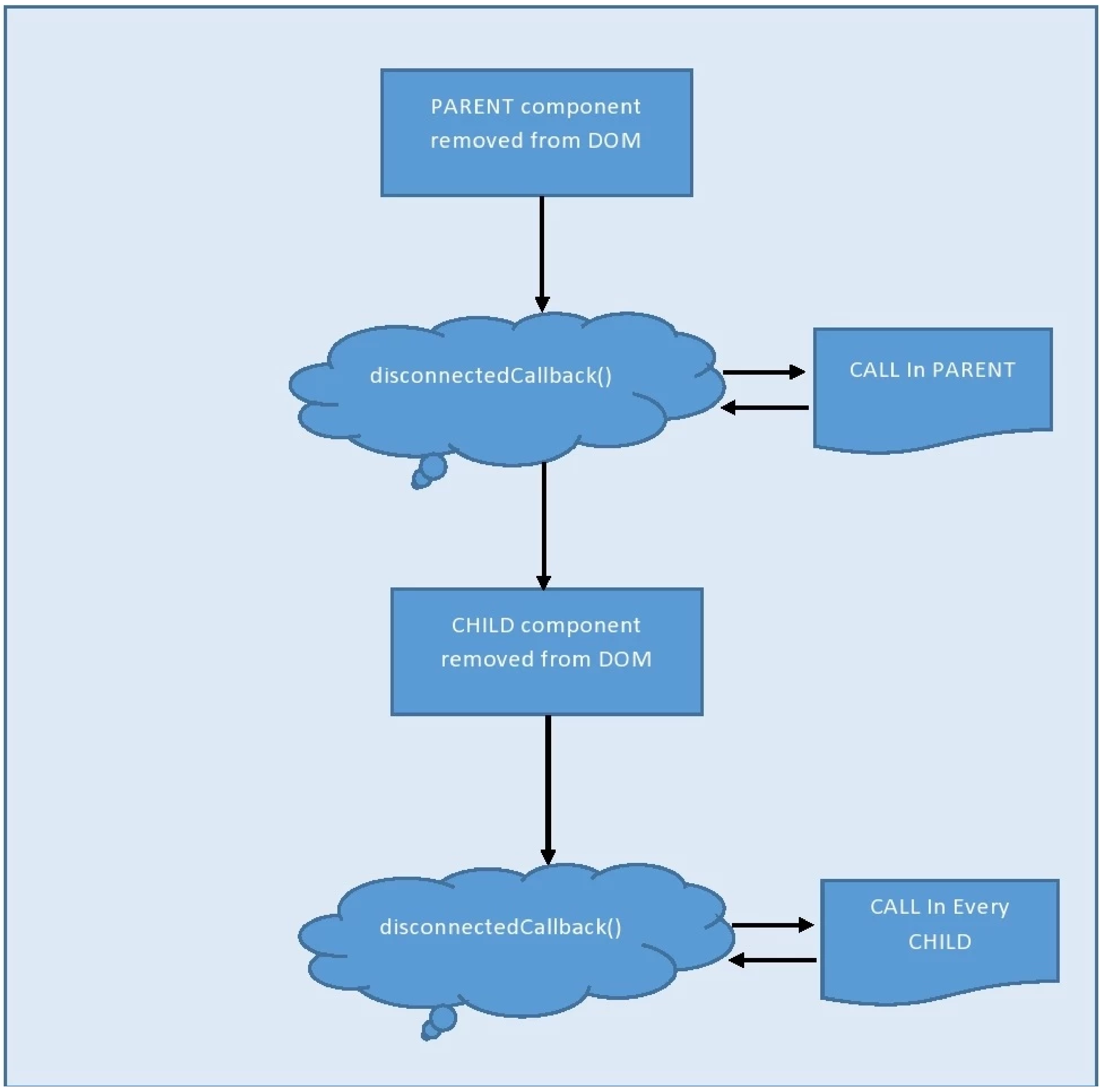
LWC Lifecycle flow from Component Creation To Render :


START
COMPONENT: PARENT
COMPONENT: CHILD
Parent Constructor() Call
Public Property Update (@api)
@api property Updated?
YES
Update @api Property
NO
Parent DOM Entry
Parent Call: connectedCallback()
Parent Re-Rendered
Parent Constructor() Call
Public Property Update (@api)
@api property Updated?
YES
Update @api Property
NO
Child DOM Entry
Child Call: connectedCallback()
Child Re-Rendered
Child Call: renderedCallback()
Parent Call: renderedCallback()
STOP

constructor() : Execute When a Component Is instantiated.
This hook in fire when a component instance is created. Child elements in the component body not exist now. Properties are not passed yet, Properties (**@api** for Public property / **@track** for Private property) are assigned to the component after this step or before the connectedCallback() hook. Don't add attributes to the host element during construction. This hook flows from parent(P) towards child (C), (P -> C) and Host element can be accessed by *this.template*.

connectedCallback() : Execute When a Component Is Injected in the DOM.
Fire when the element are injected into the DOM (subscribe). This this phase Child elements in the component body still not exist. This hook flow from Parent to Child (P -> C). To access Host element use *this.template*.

disconnectedCallback() : Execute When a Component Is Removed from the DOM.
Fire when the element is removed from a DOM (unsubscribe). This hook flows from parent to child. This hook flow from Parent to Child (P -> C).



NOTE: Browser always do garbage collects DOM events, so keep in mind not to unregister them. Otherwise its influence the flow of garbage collection process that lead to memory leaks.

render() : Execute When we need explicit call to override existing rendering functionality.
This hook explicitly use to override standard functionality, like conditionally rendering a template or importing any custom template. render() gets invoked after connectedCallback() and must return a valid HTML template.

renderedCallback() : Execute after component Render.
This hook in fire after every render of the component. This is unique functionality that present only in LWC. Such type of hook is not present in Conventional HTML custom elements specification. This hook flow from Child to Parent (C -> P). When a component re-renders, all the expressions used in the template are reevaluated.
If you use renderedCallback() to perform a one-time operation, you must track it manually like `initialRender` private property. Any activity that changes to reactive attributes, need to guard them otherwise they can trigger wasteful re-renders that may cause an infinite rendering loop.

errorCallback() : Handle Errors in descendant Component.
This hook Event is not present in Conventional HTML custom elements specification. It's a component lifecycle guard. Here this hook call when a descendant component throws an error in any of its lifecycle hooks. The `error` argument is a JavaScript native error object, and the `stack` argument is a string.
By implementing this hook one can create an error boundary component that captures errors in all the descendant components in its tree. The error boundary component can log stack information and provide an alternative view of information to the users about the error details and what to do next. It's similar to *catch{}* block.

NOTE: Note that an error boundary component catches errors only from its children, not from itself.