Differential loading

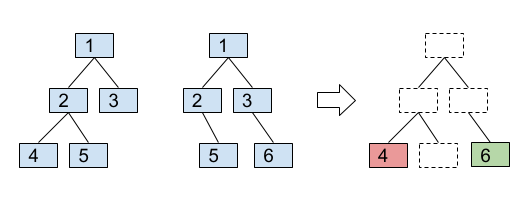
Your Angular 8 apps will now be automagically more performant, thanks to differential loading.

With differential loading, two bundles are created when **building** for production:   
a bundle for modern browsers that support ES2015+ and   
a bundle for older browsers that only support the ES5 version of JavaScript.

**The correct bundle will be loaded automatically** by the browser, thanks to the support of ES6 modules in newer browsers.

How Virtual DOM Works

React was the first mainstream framework to use virtual DOM, which is defined by this key idea:

Every component creates a new virtual DOM tree every time it gets rerendered. React compares the new virtual DOM tree with the old one and then applies a series of transformations to the browser DOM to match the new virtual DOM tree. 

Virtual DOM has two main advantages:

We can use any programming language to implement the component’s render function, so we don’t need to compile anything. React developers mainly uses JSX, but we can use plain JavaScript as well.

We get a value as a result of rendering component. It can be used for testing, debugging, etc..

Opting into Angular Ivy

IVY is an initiative to build a next-generation rendering pipeline for Angular, and, for this, the Angular team is currently rewriting the code that translates the Angular template to whatever we rendered in the browser. It uses the incremental DOM

Incremental DOM

Incremental DOM means every component is compiled with a series of instructions that creates the DOM tree and updates them when data changes.

There Are Mainly Two Key Concepts for IVY

* Tree Shakable: Remove unused code so the application only pays attention to the code it’s using, hence a smaller bundle and faster run time
* Local: To recompile only the components that are changing which would result in a faster compilation

.This preview brings features like:

* Smaller bundle size.
* Faster re-build time resulting in increased performance.
* Cleaner code, leading to easier debugging.
* Improved template type-checking.
* Great Backwards compatibility.
* Improved payload size for improvement in real-world applications.

F zman kan l modo3 mashy keda :-



the Template Data:-

is a **view** definition produced by the compiler. The definition holds metadata relevant to a component and is used as a blueprint when creating a component view

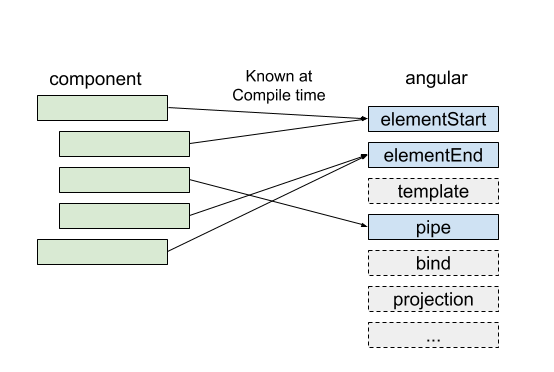
w dlw2ty b3d l incremental Dom l modo3 b2a mashy keda



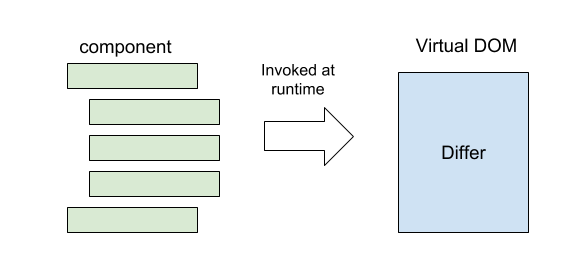
template instructions:-

de feha l instruction l ms2ola 3n create and update l component if data changed

When using incremental DOM, the framework does not interpret the component. Instead, the component references instructions. If it doesn’t reference a particular instruction, it will never be used. And since we know this at compile time, we can omit the unused instruction from the bundle.

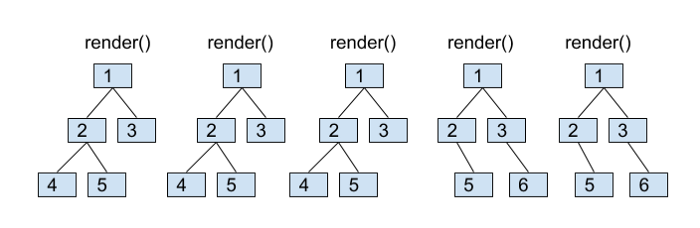


Virtual DOM requires an interpreter. What part of that interpreter is needed and what part is not isn’t known at compile time, so we have to ship the whole thing to the browser.



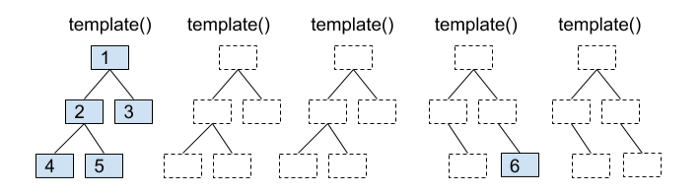
Why Incremental DOM Has Low Memory Footprint?

Virtual DOM creates a whole tree from scratch every time you rerender.



Incremental DOM,

on the other hand, doesn’t need any memory to rerender the view if it doesn’t change the DOM. We only have to allocate the memory when the DOM nodes are added or removed. And the size of the allocation is proportional to the size of the DOM change.



Since most of render/template calls don’t change anything (or change very little), this can result in huge memory savings.