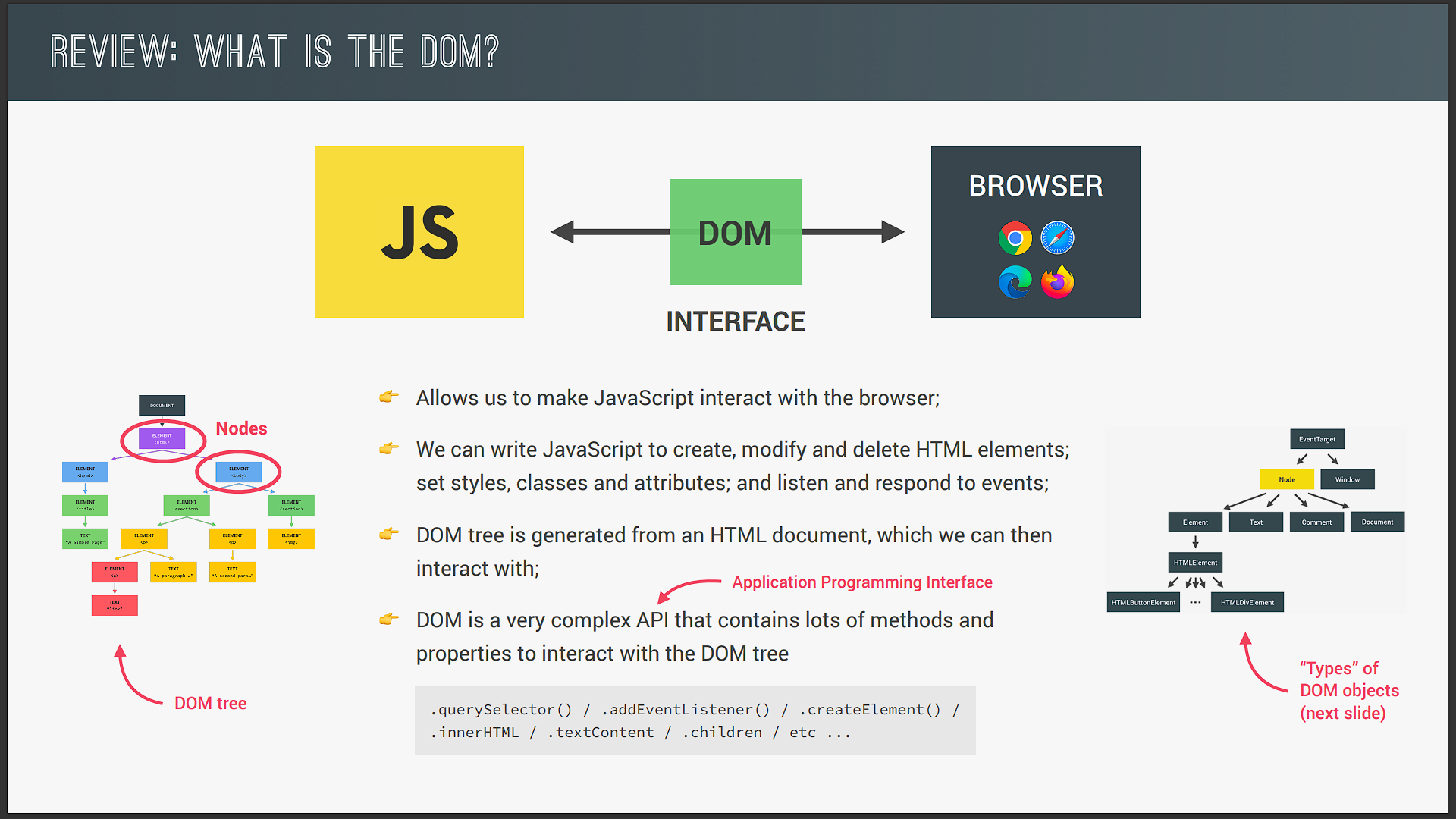
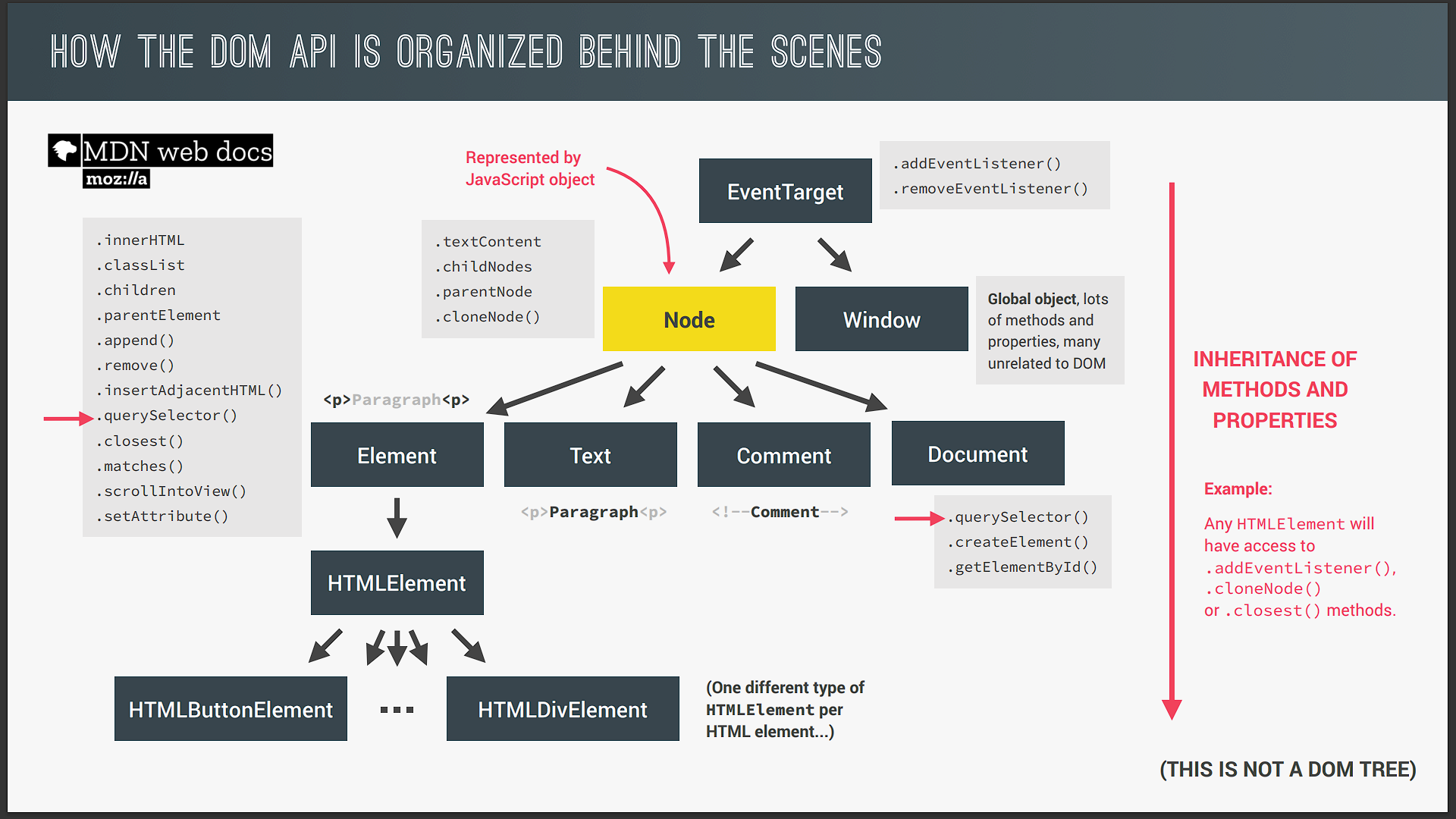
Let's start this section by learning how the DOM really works behind the scenes and more specifically how the DOM is organized internally. So first remember that the DOM is basically the interface between all JavaScript code and the browser or more specifically HTML documents that are rendered in and by the browser.

So let's remember what we already know about the DOM which is that we can use it to make JavaScript interact with the browser and again more specifically we can create and modify and delete elements set styles and classes and attributes and listen and respond to events. In practice this works because a DOM tree is generated from any HTML document and a DOM tree is a tree like structure made out of nodes which looks something like this.



And we can then interact with this tree as we already did a couple of times in this course. Now how does that interaction actually work? Well the DOM is a very complex API which remember stands for application programming interface. So it's the interface we can use to programmatically interact with the DOM. In practice that means that the DOM contains a ton of methods and properties that we use to interact with the DOM tree such as the querySelector addEventListener or createElement methods or also the innerHTML textContent or children properties and many many more.

Now in the DOM there are different types of nodes just as I mentioned before. For example some nodes are HTML elements but others are just text remember? And this is really important to understand because all these DOM methods and properties are organized into these different types of objects. And so let's now take a look at how the DOM API is organized behind the scenes.



So first every single note in the DOM tree is of the type node. And such as everything else in JavaScript each node is represented in JavaScript by an object. This object gets access to special node methods and properties such as text content child nodes parent nodes clone nodes and many others. Now we already know that there are different types of nodes. Right? So how should these be represented? Well this node type has a couple of child types so to say. And these are the element type the text type the comment type and also the document type. So whenever there is text inside any element we already know that it gets its own node. Right? And that node will be of the type text. And the same actually happens for HTML comments and that's because the rule is that everything that's in the HTML has to go into the DOM as well. Now for the element itself there is the element type of node. And this type of node gives each HTML access to a ton of useful properties such as innerHTML classList children or parent element. There are also many useful methods like append remove insertAdjacentHTML querySelector closest and that's just to name a few.

So again each element will be represented internally as an object. Now just to make this complete the element type has internally an HTML element child type. And that element type itself has exactly one child type for each HTML element that exists in HTML. So we have a special type for buttons a special type for images for links and so on and so forth. And that's important because each of these HTML elements can have different unique properties. For example and image has a source attribute in HTML which no other element has. Or the anchor element for links has the HREF attribute which also no other element has. And so the DOM needs a way of storing these different attributes and therefore different types of HTML elements were created in the DOM API.

And just to make sure that we're all on the same page here this diagram that I'm showing you here is of course not a DOM tree. Right? So this is not a representation of any HTML document. This is just a way that different types of nodes are represented behind the scenes in the DOM API. Now right? But anyway now comes the really important part because what makes all of this work is something called inheritance. So what is inheritance?

Well inheritance means that all the child types will also get access to the methods and properties of all their parent node types. For example an HTML element will get access to everything from the element type like innerHTML or classList or all these other methods and properties. And besides that it will also get access to everything from the node type because that is also its parent type. Okay? So we can think of this as though the HTML button element for example is also an element and also a node. All right? Now this might seem all a bit weird and confusing but don't worry. We will learn why this kind of inheritance works very soon when we finally talk about object oriented JavaScript.

For now what I want you to understand is that a DOM API is broken up into these different types of nodes. I also want you to understand that each of these types of nodes has access to different properties and methods and that some of them even inherit more properties and methods from their ancestors in this organization. All right?

Now we didn't talk yet about the documents node type. So document which we use all the time in DOM manipulation is in fact just another type of node so it contains important methods such as querySelector createElement and getElement by I.D. And note how querySelector is available on both the document and element types. So keep this in mind because it will be important later on. All right and now there is just one final missing piece here because the DOM API actually needs a way of allowing all the node types to listen to events. And remember we usually listen for events by calling the addEventListener method on an element or the document. Right? So why does that actually work? Well its because there is a special node type called EventTarget which is a parent of both the node type and also the window node type. And so with this thanks to inheritance we can call addEventListener on every single type of node in the DOM API because all elements as well as document and window and even text and comment will inherit this method and therefore we will be able to use addEventListener on all of them just as if it was their own method. Now just to be clear we do never manually create an eventTarget object. Okay. This is just an abstract type that we do not use in practice. This all really happens behind the scenes to make all the functionality work as we expect it to work.

So in a nutshell this is how the DOM API works and is structured behind the scenes. There are still some simplifications here but this is all that really matters. And I really wish that I could have had this diagram when I learnt JavaScript for the first time. Because I really think this helps structuring all this information in your mind. Now if you want to go even deeper than this there is still tons of material that you can check out in the MDN documentation and if you ask me it's all really fascinating. But again all that you need to know is really in this lecture. It took me a lot of hours to put this one together but I think it was well worth it and I hope you think the same. But anyway lets now move on to the practical part Of this section were we will then finally use many of these things in action.