DOM vs. Canvas

WE'LL COVER THE FOLLOWING

- Retained Mode (DOM)
- Immediate Mode (Canvas)

Did you know that you have a choice in how you can get things to display in your browser? Well...you do! The most common approach for getting content to display on your screen is by working with DOM elements. This is where you (and 99% of the entire world) create HTML, CSS, and JavaScript and have elements appear magically. The other approach uses the `canvas` element. With a `canvas`, you manually specify exactly what you want drawn and displayed on your screen. Both of these approaches have their uses. For the kinds of visually complex applications you will be creating, knowing when to use which is a good thing for you to be aware of.

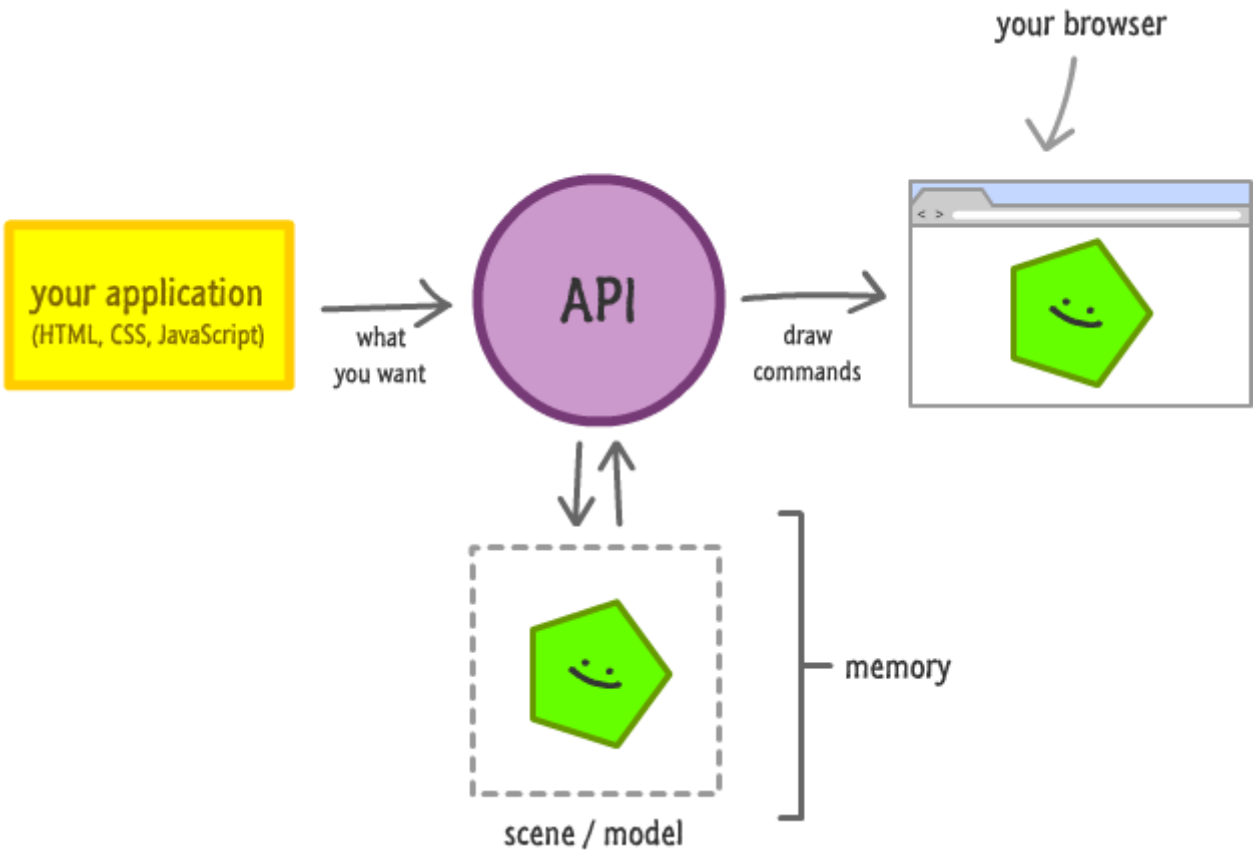
To help you with this, let's take a step back. In fact, let's take many steps back and look at how our two approaches map to how your browser translates what you want into something that gets displayed. Core to this translation are two modes called **Retained Mode** and **Immediate Mode**. While it may not seem like it right now, understanding the details of both of these modes will help you know when to rely on the DOM APIs and when to use `canvas` to display visuals on the screen.

Onwards!

Retained Mode (DOM)

In a retained mode system, the way you get things to display on the screen is by sending your hopes, dreams, and desires to your browser's Graphics API. This API, much like Santa Claus, gives you whatever you ask for.

The following diagram roughly describes the division of labor between you, the Graphics API, and your browser:



The yellow box represents the HTML, CSS, and JavaScript that makes up your application. While you may not have thought about your markup and code in

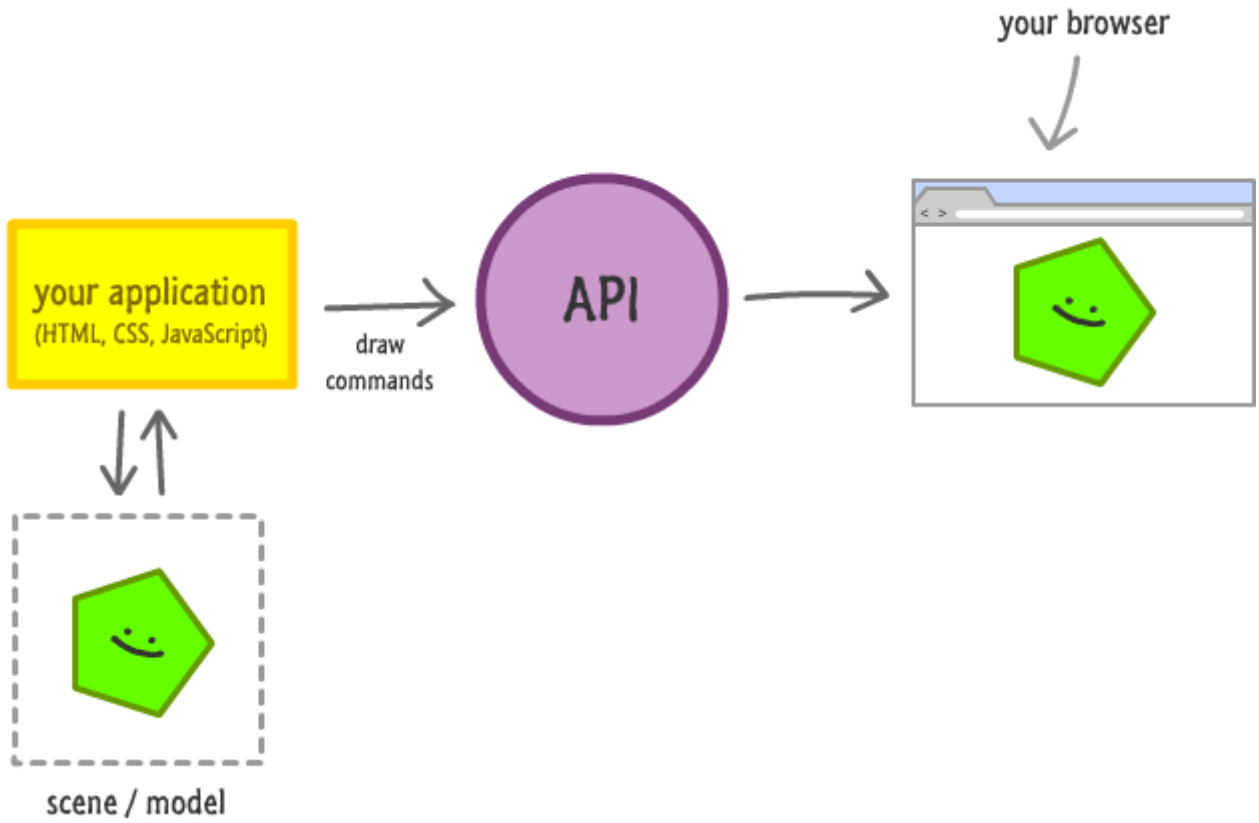
quite this way, almost everything you specify is nothing more than a drawing instruction that tells your browser what to display on the screen.

This translation between your raw markup and code to something visual is handled by your browser's Graphics API. This API takes what you specify and creates an in-memory model (often referred to as a ** *scene, object list* ** or ** *display list* **) of what the final output should look like. Once it creates this model, the final step is to translate the model into the arcane draw commands that your browser understands.

All of this happens very seamlessly behind the scenes. You simply define your application. The details of getting what you defined into something you can see is automatically handled for you.

Immediate Mode (Canvas)

Contrasting the gentle comforts of a retained mode system is the immediate mode one where...well, let's just look at the diagram first:



In an immediate mode system, you do all of the heavy lifting. You not only specify what needs to be drawn, you also create and maintain the model as well. As if all of this wasn't enough, you also specify the draw commands to get your browser to actually update! Your browser's Graphics API, the API that did so much for you in the retained mode world, doesn't do much here. It simply takes your draw commands and sends them off to the browser for

simply takes your draw commands and sends them off to the browser for execution.

In HTML, immediate mode comes into play when you are using the `canvas` element. Anything you wish to draw inside it requires you to issue primitive draw commands and handle all aspects of managing the scene and redrawing when changes happen.