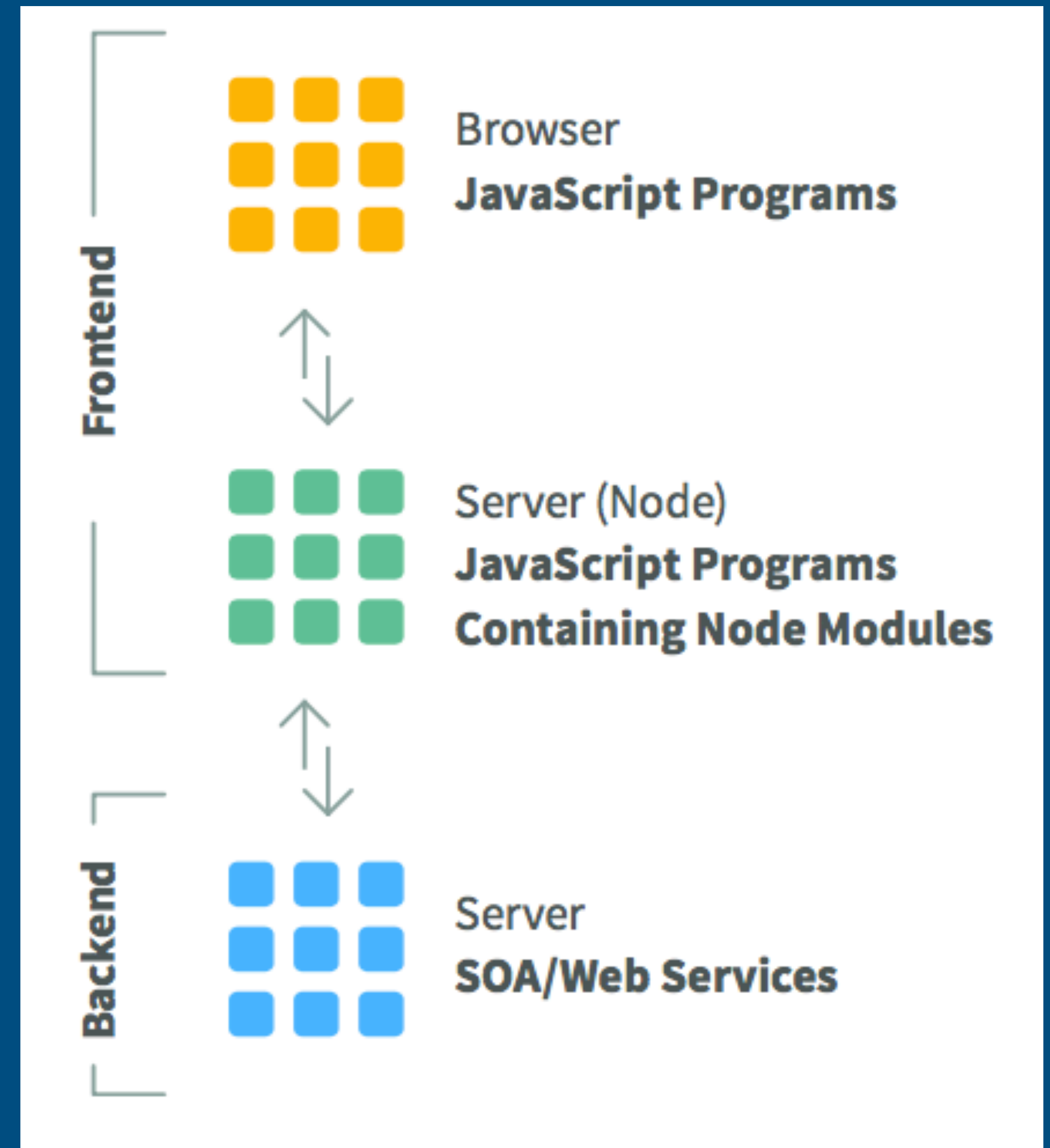


Node Context



Full Stack Web Development

Scope

- JavaScript on the Server
- Asynchronous Programming
- Module-driven Development
- Small Core, Vibrant Ecosystem
- The 'Frontend Backend'



JavaScript on the Server



- Large JavaScript developer community already building sophisticated web applications.
- JavaScript consistently ranks among the most popular languages. JavaScript seen as approachable (“Developer joy”)
- JavaScript on the server with Node further establishes it as the language of the web.
- Its new uses on the server are helping to shape the future of the language itself.

Synchronous Programming : Java

- The default mode for resource access in Java in synchronous
- Each database access in this example (Play/Java) is an example of a “Blocking” call.
 - controller process suspended
 - external database service contacted & request made
 - database responds with data
 - controller process ‘woken up’ and it moves on to next instruction...

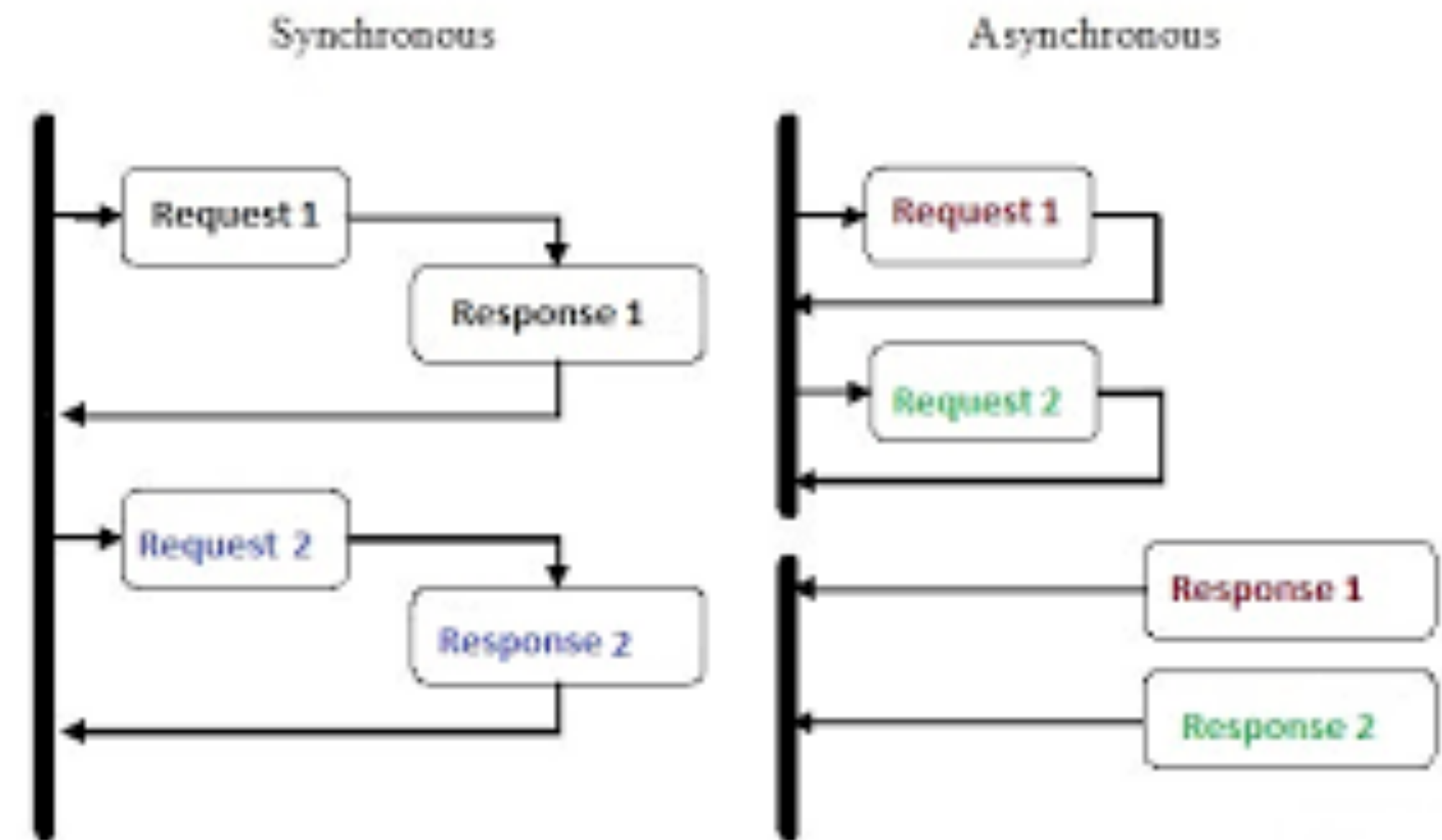
```
public class PlaylistCtrl extends Controller
{
    public static void index(Long id)
    {
        Playlist playlist = Playlist.findById(id);
        Logger.info ("Playlist id = " + id);
        render("playlist.html", playlist);
    }

    public static void deletesong (Long id, Long songid)
    {
        Playlist playlist = Playlist.findById(id);
        Song song = Song.findById(songid);
        Logger.info ("Removing" + song.title);
        playlist.songs.remove(song);
        playlist.save();
        song.delete();
        render("playlist.html", playlist);
    }

    public static void addSong(Long id, String title,
                               String artist, int duration)
    {
        Song song = new Song(title, artist, duration);
        Playlist playlist = Playlist.findById(id);
        playlist.songs.add(song);
        playlist.save();
        redirect ("/playlists/" + id);
    }
}
```

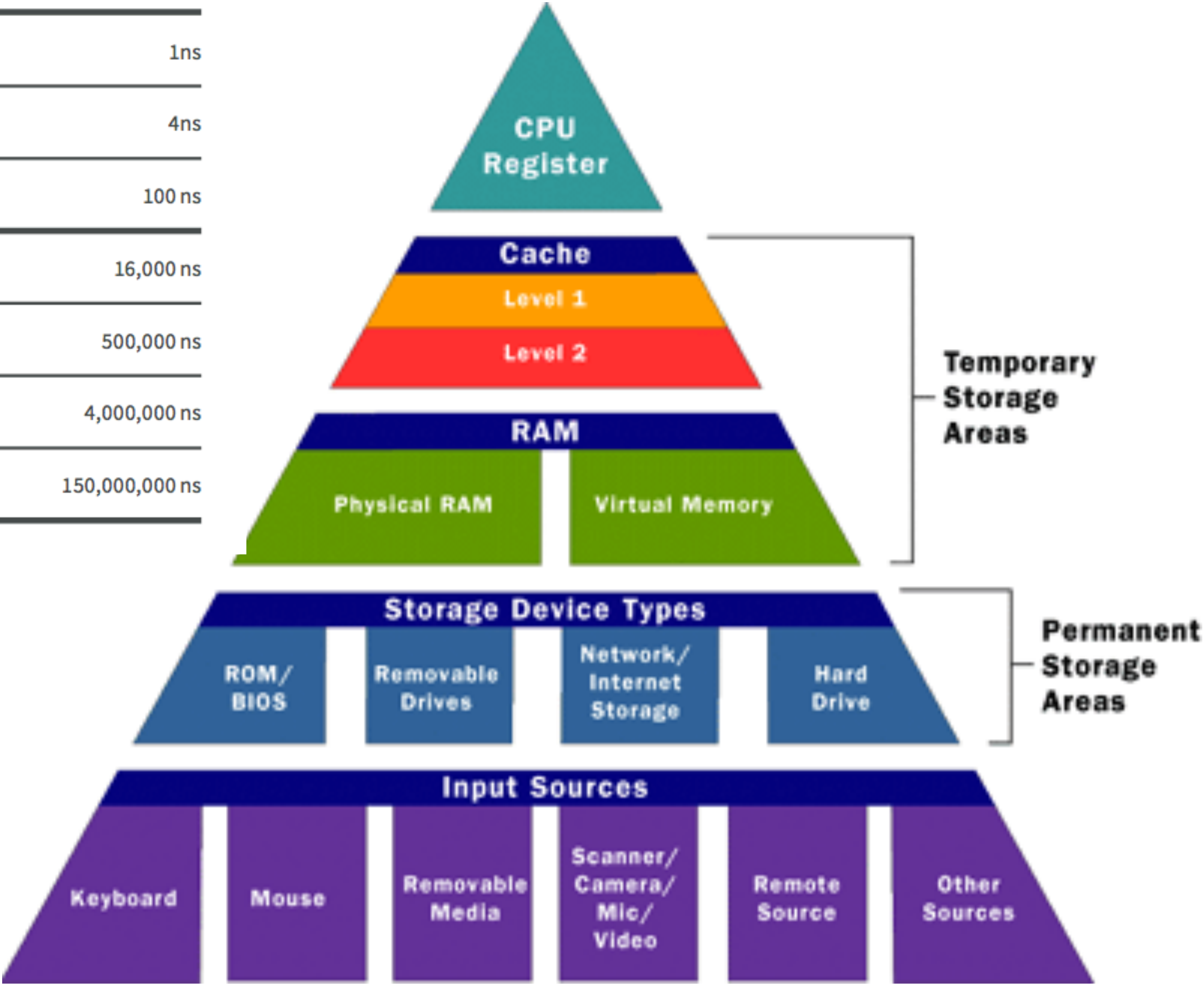

Asynchronous Programming I

- JavaScript was designed for the web and the browser, where nothing is synchronous.
- Asynchronous JavaScript programs can perform many complex, parallel tasks in the browser.
- Node takes asynchronicity to the extreme on the server, making it the perfect choice for I/O-heavy and highly concurrent applications.
- Applications built with Node are built for predictable scalability - design patterns adopted within Node programmes confer robust scalability without the overhead required by complicated synchronization mechanisms



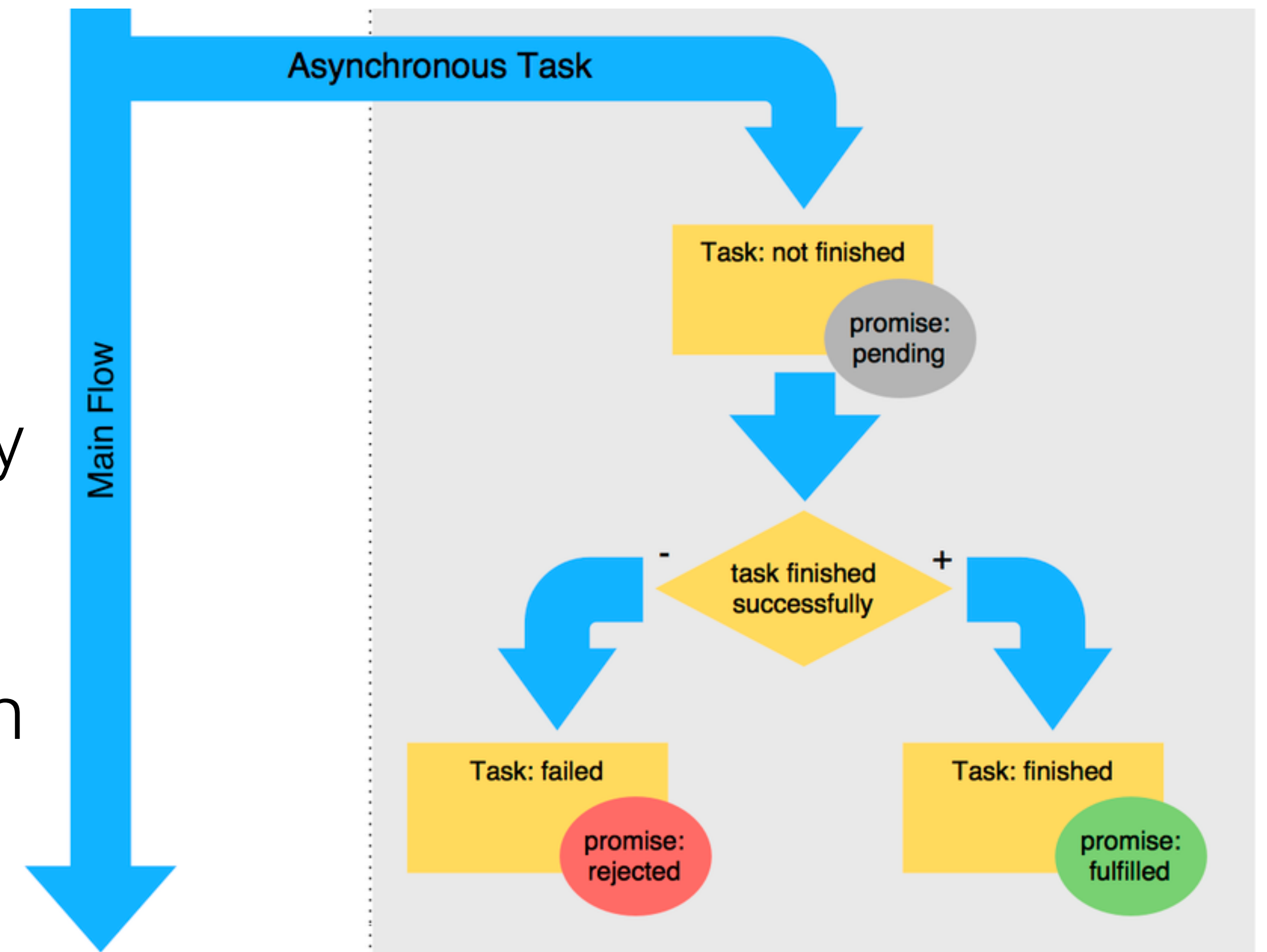
I/O Is Expensive

CLASS	OPERATION	TIME COST
MEMORY	L1 Cache Reference	1ns
	L2 Cache Reference	4ns
	Main Memory Reference	100 ns
I/O	SSD Random Read	16,000 ns
	Round-trip in Same Datacenter	500,000 ns
	Physical Disk Seek	4,000,000 ns
	Round-trip from AU to US	150,000,000 ns



Asynchronous Programming II

- Node requires developers to embrace a different mindset in the form of asynchronous programming
- By treating I/O as a special class of operation, developers must design highly performant applications by default.
- Node is single-threaded by nature, which is embraced as a part of the application design.



Module-driven Development I

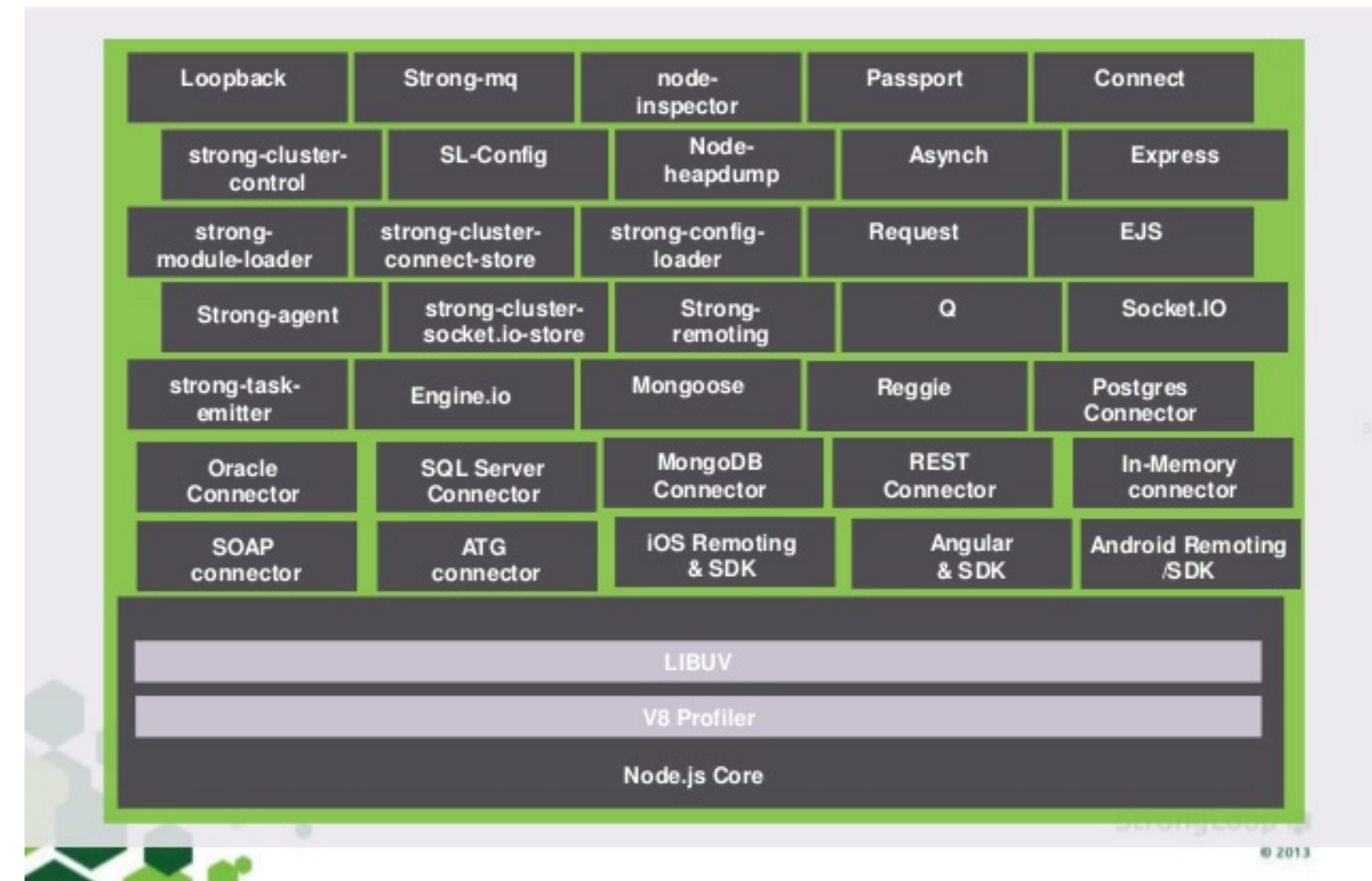


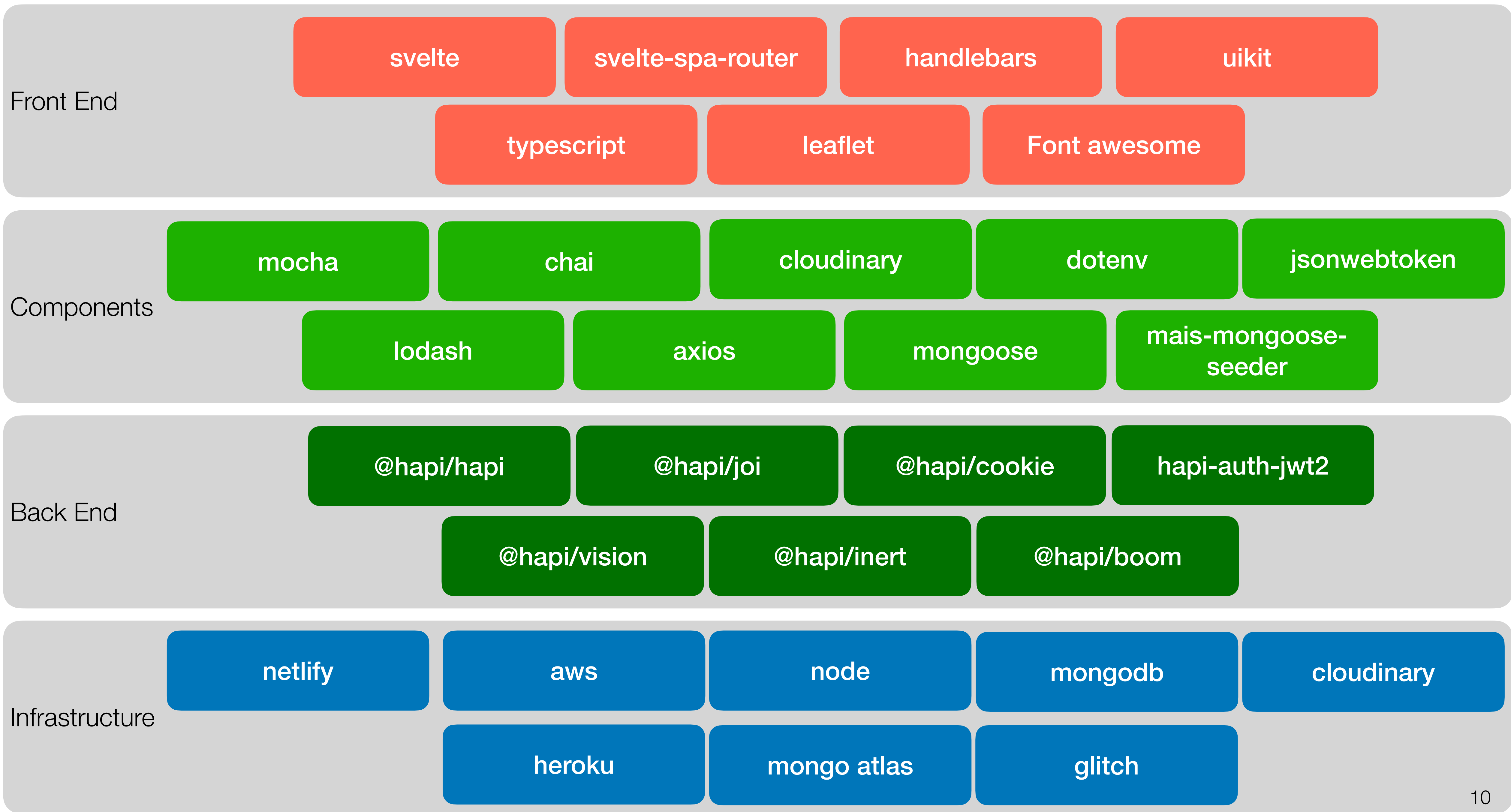
- Node is modular by nature.
- Node embraces a practice of “Throw-awayability” becoming pervasive in the services oriented software design world - i.e. encourage developers to think in terms of creating small services that can be easily replaced or updated when necessary.
- By adopting a module-driven approach, Node developers can deconstruct the functionality of large monolithic applications and redesign them as a series of Node modules, bundled together to form a collection of services.
- This establishes an elegant simplicity in building scalable application functionality that improves both business and developer agility and leads to greater code-reuse.

Module-driven Development II

- Having development teams focusing on developing modules enables them to:
- Maintain focus on essential functionality
- Better test, validate and document that functionality
- More easily share and collaborate with other teams

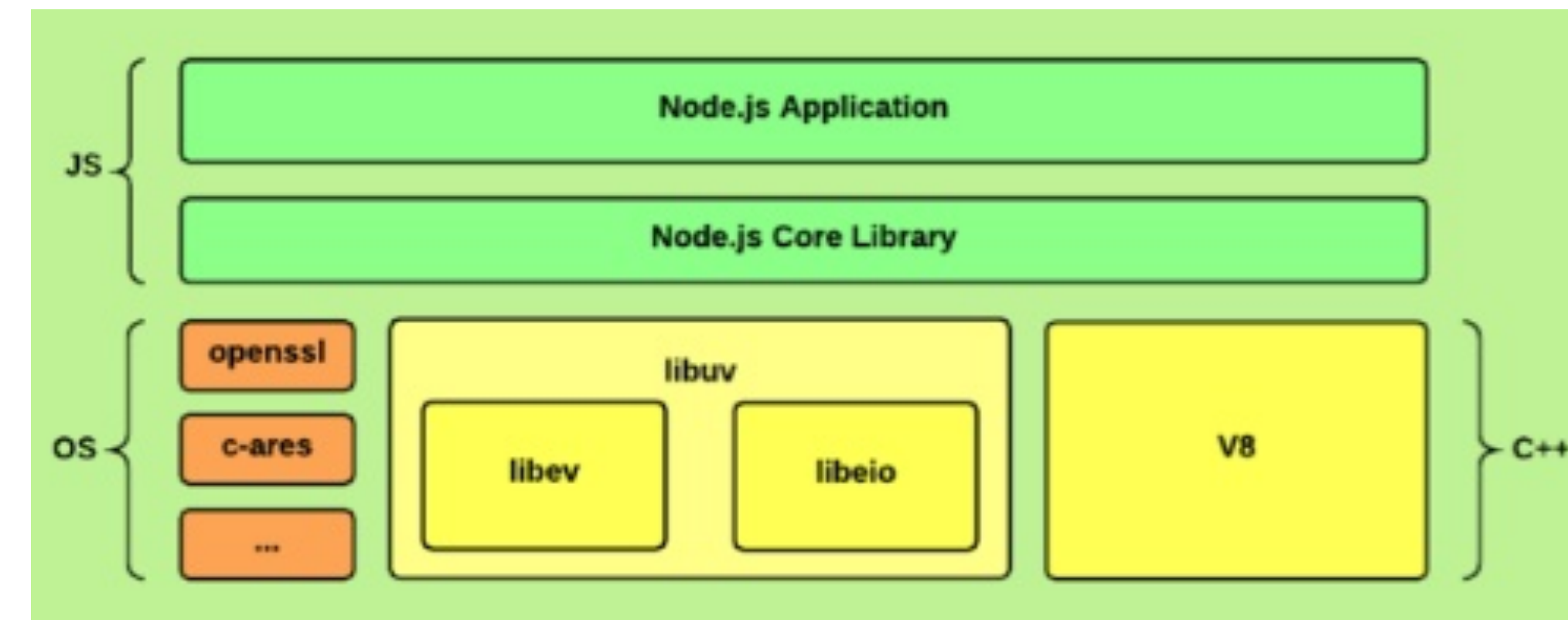
Node.js modules (Sample set)



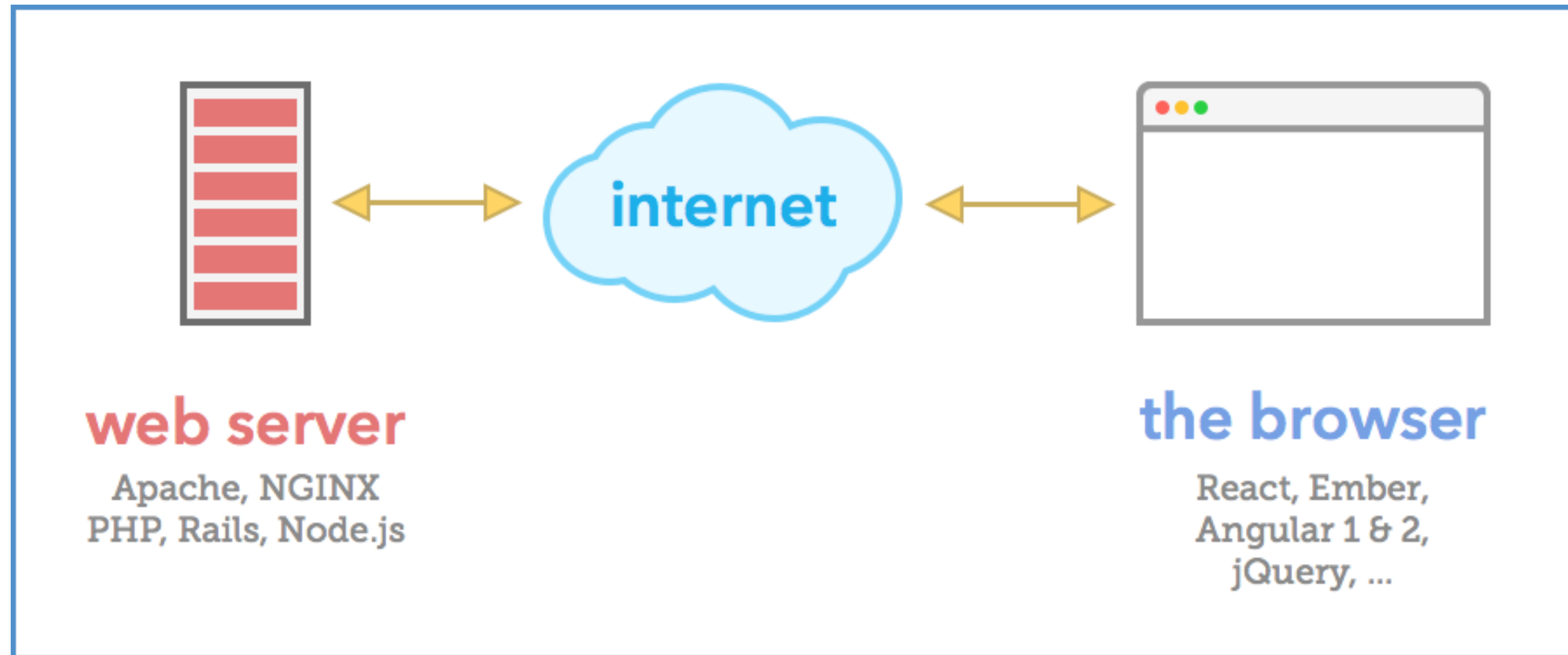


Small Core, Vibrant Ecosystem

- Large monolithic applications are often subject to “mission creep” of applications and the development environments that build them. Over time this results in feature rich but bloated products.
- Node avoids this scenario by creating a small core of essential functionality that is studiously defended and constantly debated by the Node community. This pushes experimentation to the edges and encourages a vibrant ecosystem and development culture.
- This ethos also extends to Node- style development, with developers constantly thinking about how to keep modules small and focused and splitting apart functionality where the “do one thing well” rule is broken.



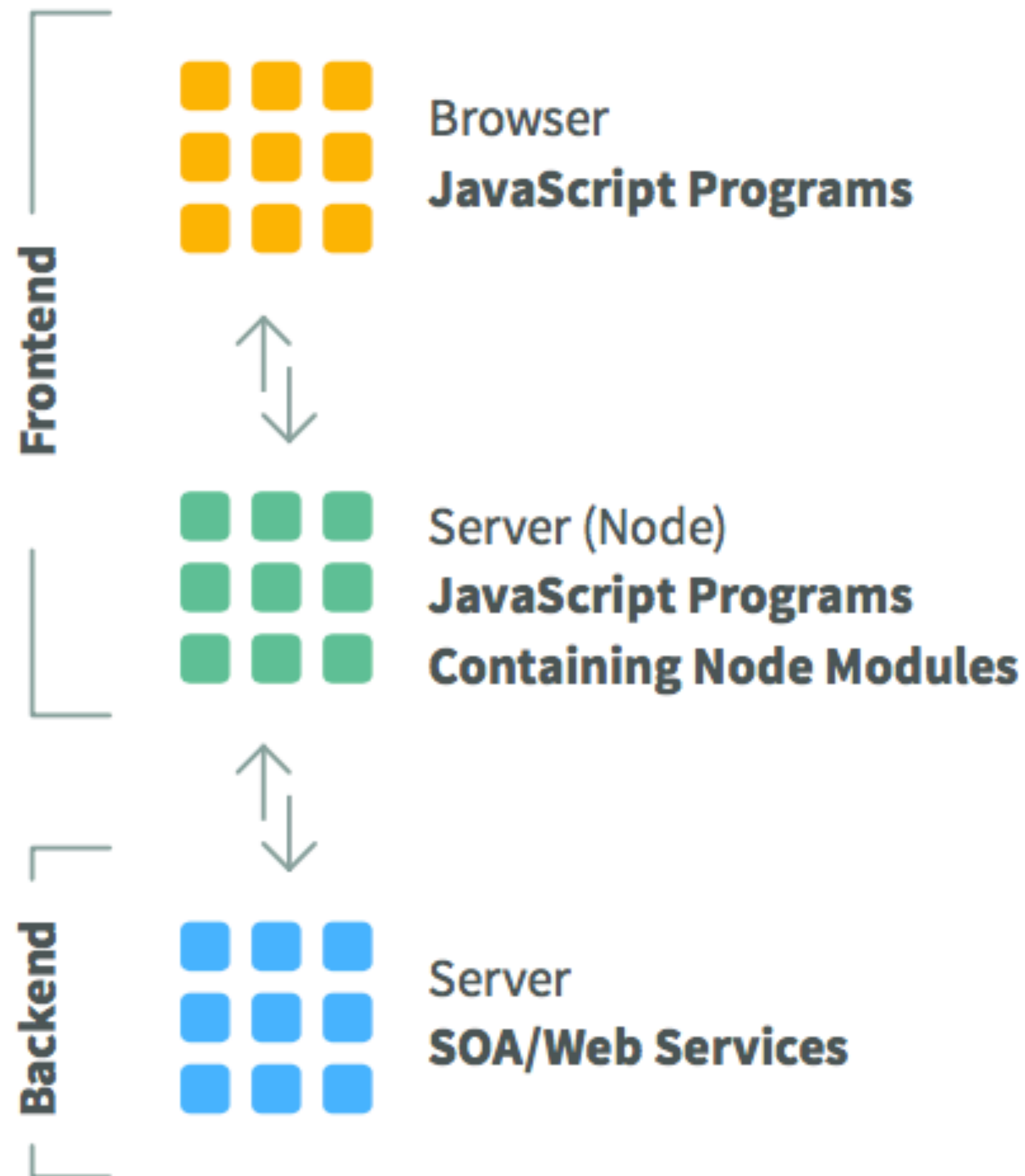
Traditional Frontend / Backend



The “Frontend Backend”

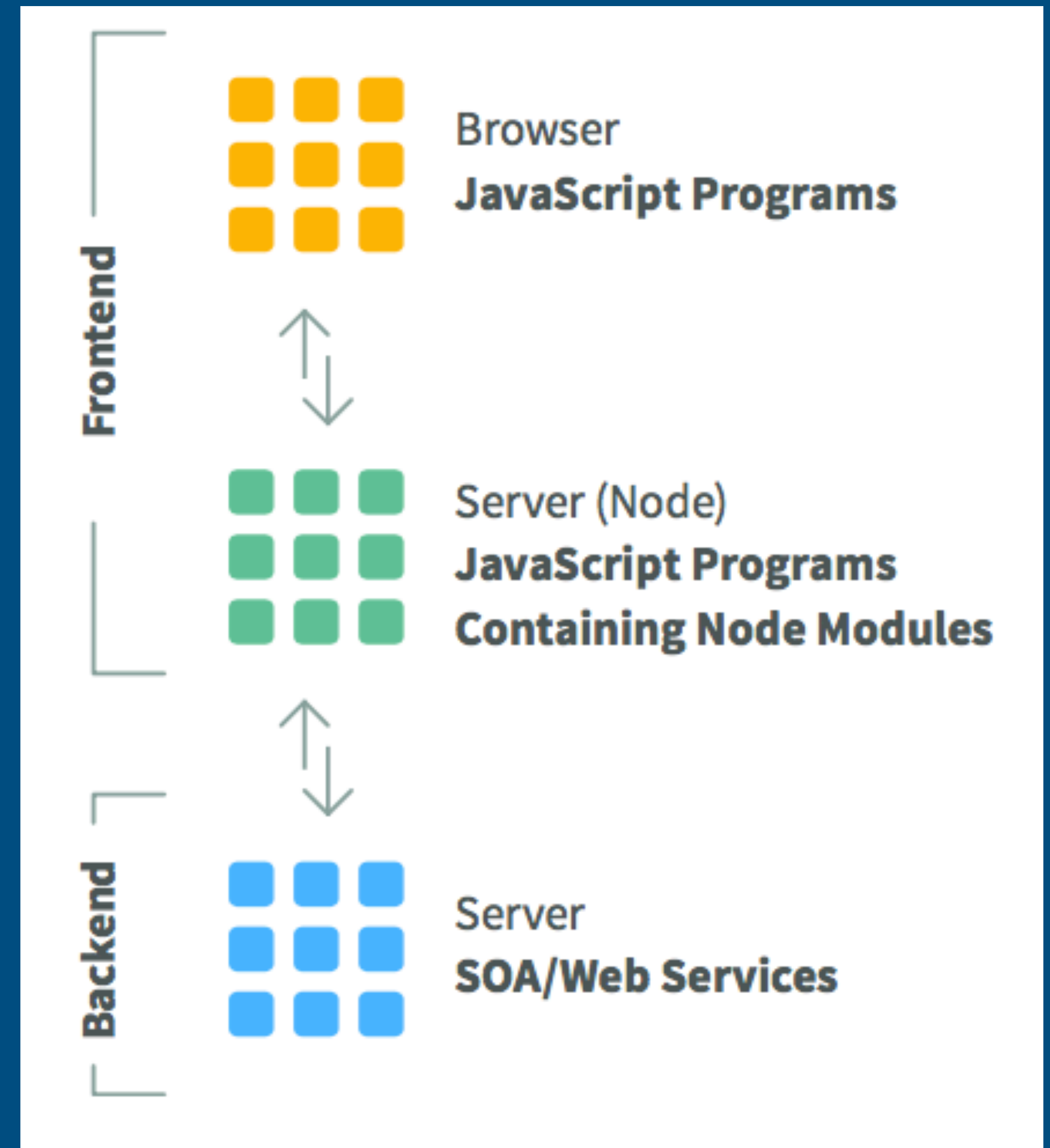
- Rich client teams who have been building exciting, dynamic JavaScript experiences have run up against problems from building large, monolithic structures that naturally result from traditional top-down programming.
- The result is poor performance and scalability and frustration for end users.
- Front end-developers must also rapidly iterate on the customer experience to keep users engaged.
- This has lead to the growth if the ‘Frontend Backend’ pattern, with node as a clear choice.

The Frontend Backend



- The application frontend needs a lightweight, dynamic back-end to deliver the scale and response times needed.
- The Frontend Backend is architectural tier added to a system to specifically serve frontend resources (templates, html, css , etc.) in front of a legacy system or API service tier.
- A frontend backend empowers frontend teams to quickly evolve the user experience to respond to rapidly changing conditions on the ground – such as news items, social happenings, and sporting or cultural events – all while being able to think and operate in the familiar JavaScript mindset.

Node Context



Full Stack Web Development