TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS (502070)

NODEJS OVERVIEW

TDTU

- Node.js is an **open-source**, **cross-platform JavaScript runtime** environment and library for running web applications outside the client's browser.
- Developers use Node.js to create **server-side web applications**, and it is perfect for **data-intensive applications** since it uses an **asynchronous, event-driven model**.

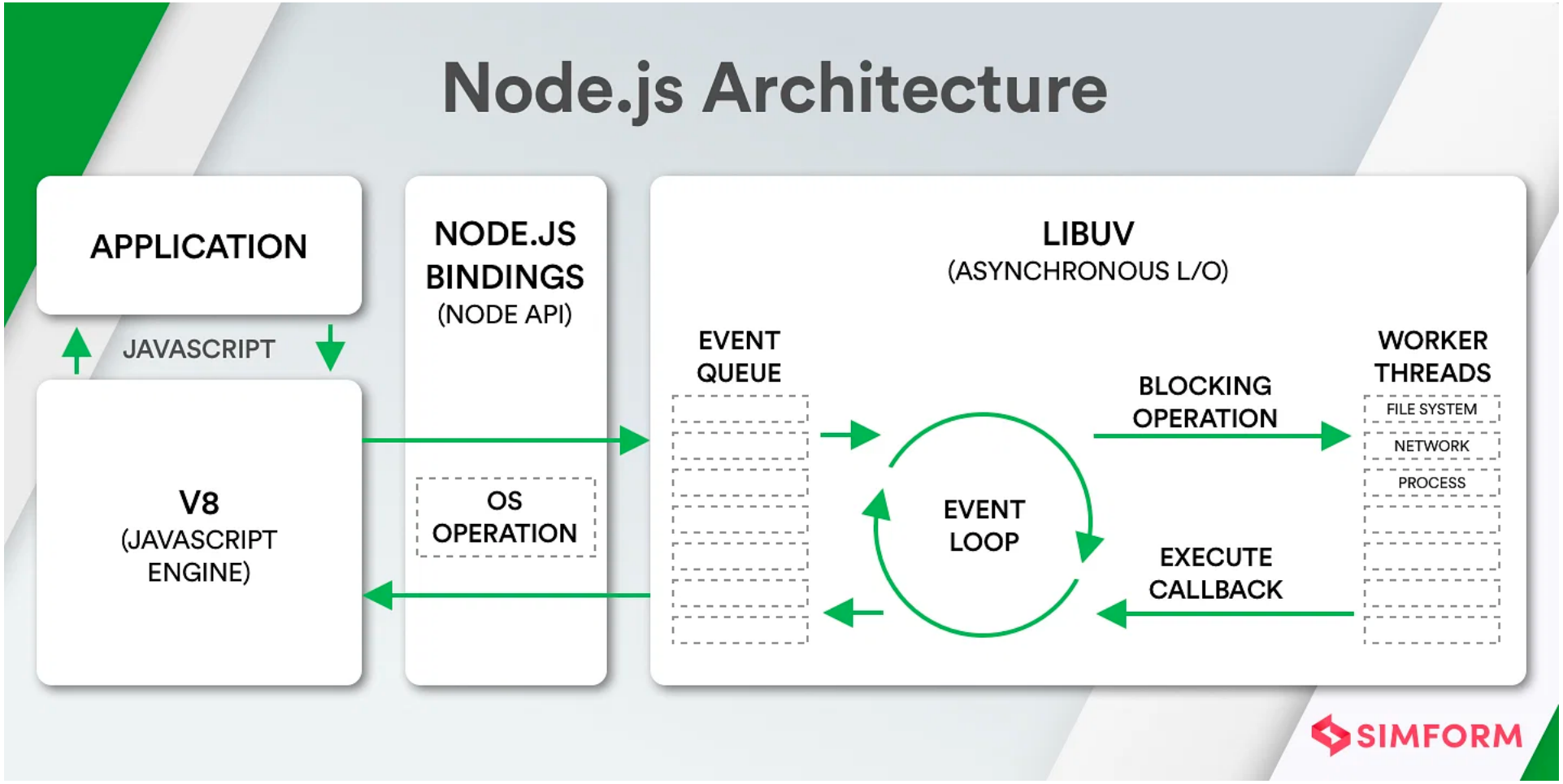
Why Nodejs?

- Built on Google Chrome's V8 engine, so its execution time is very fast and it runs very quickly.
- > 50,000 bundles available in the Node Package Manager
- No need to wait for an API to return data, totally asynchronous, totally non-blocking.
- Loading time reduced due to better synchronization of the code between the client and server for having the same code base.
- Open source, JavaScript framework code based.

Nodejs features

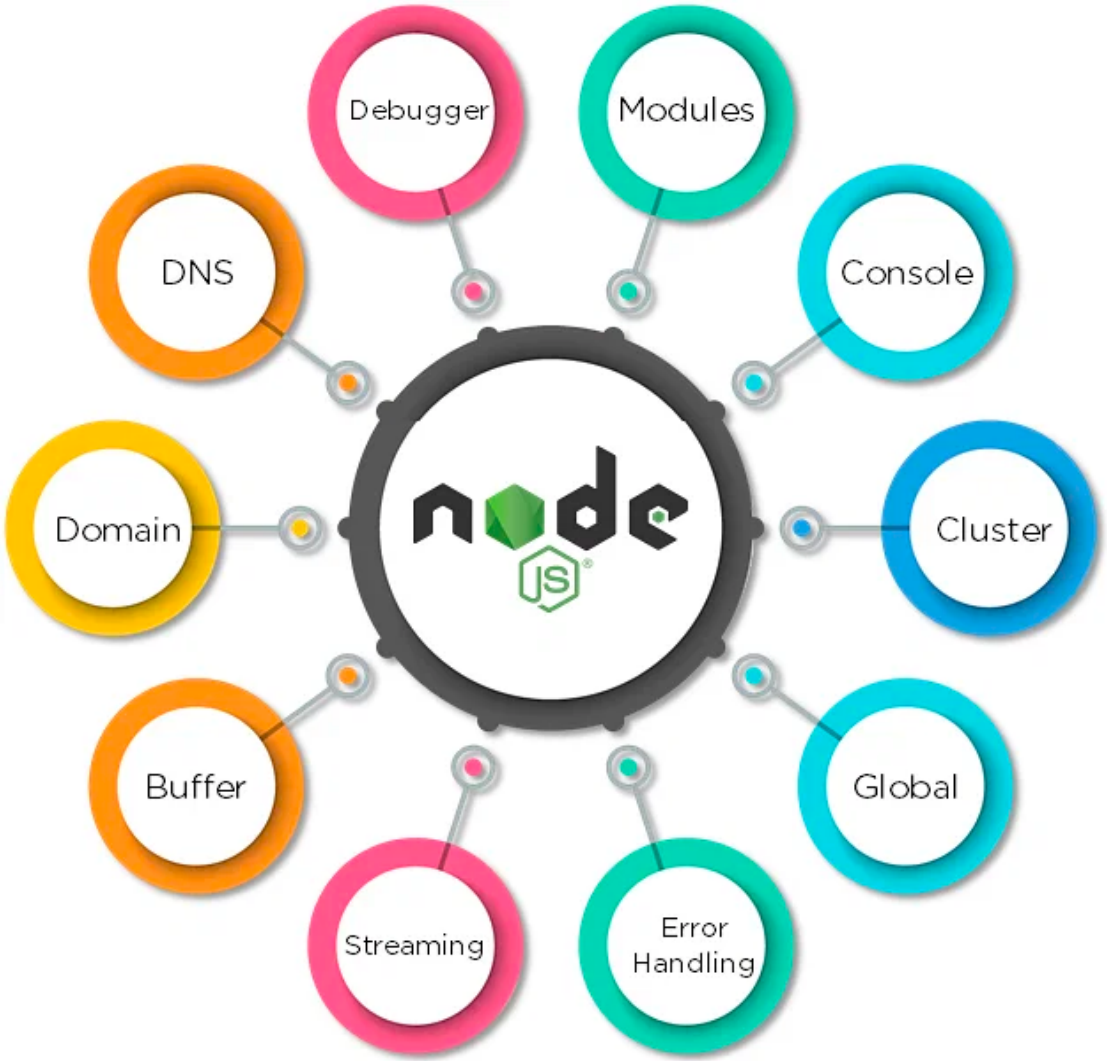
- ✓ **Asynchronous in Nature and Event driven**: Without waiting for the data from the API, it directly moves to the next API
- ✓ **Single Threaded Architecture**: With event looping, event driven mechanism, the NodeJS servers reply in a non-blocking or an asynchronous manner
- ✓ **Scalable**: Partition the applications horizontally and this partition procedure is mainly achieved by it due to the use of child processes
- ✓ **Quick Execution time for code**: V8 JavaScript runtime motor
- ✓ **Compatibility on the cross platforms**: Windows, UNIX, LINUX, MacOS and other mobile devices
- ✓ **Uses JavaScript**
- ✓ **Fast Data Streaming**: The files are processed and uploaded simultaneously by NodeJs
- ✓ **No Buffering**: The data is never buffered in NodeJs application.

Nodejs architecture



Source: simplilearn

Nodejs components



Source: simplilearn

Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. In order to include a module in a Node.js application, use the require() function with the parenthesis containing the name of the module.

```
// CREATING A WEB SERVER

// Include modules
var http  = require('http');
var server =
http.createServer(function(req, res){
  //write your code here
});
server.listen(2000);
```

Nodejs components – module

Node.js has many modules that provide the basic functionality needed for a web application.

Core Modules	Description
http	Includes classes, methods and events to create Node.js http server
util	Includes utility functions useful for developers
fs	Includes events, classes, and methods to deal with file I/O operations
url	Includes methods for URL parsing
querystring	Includes methods to work with query string
stream	Includes methods to handle streaming data
zlib	Includes methods to compress or decompress files

Nodejs components – console

The console is a module that provides a method for debugging that is similar to the basic JavaScript console provided by internet browsers. It prints messages to **stdout and stderr**.

```
// WRITING "hello world" to console
console.log('hello world');
```

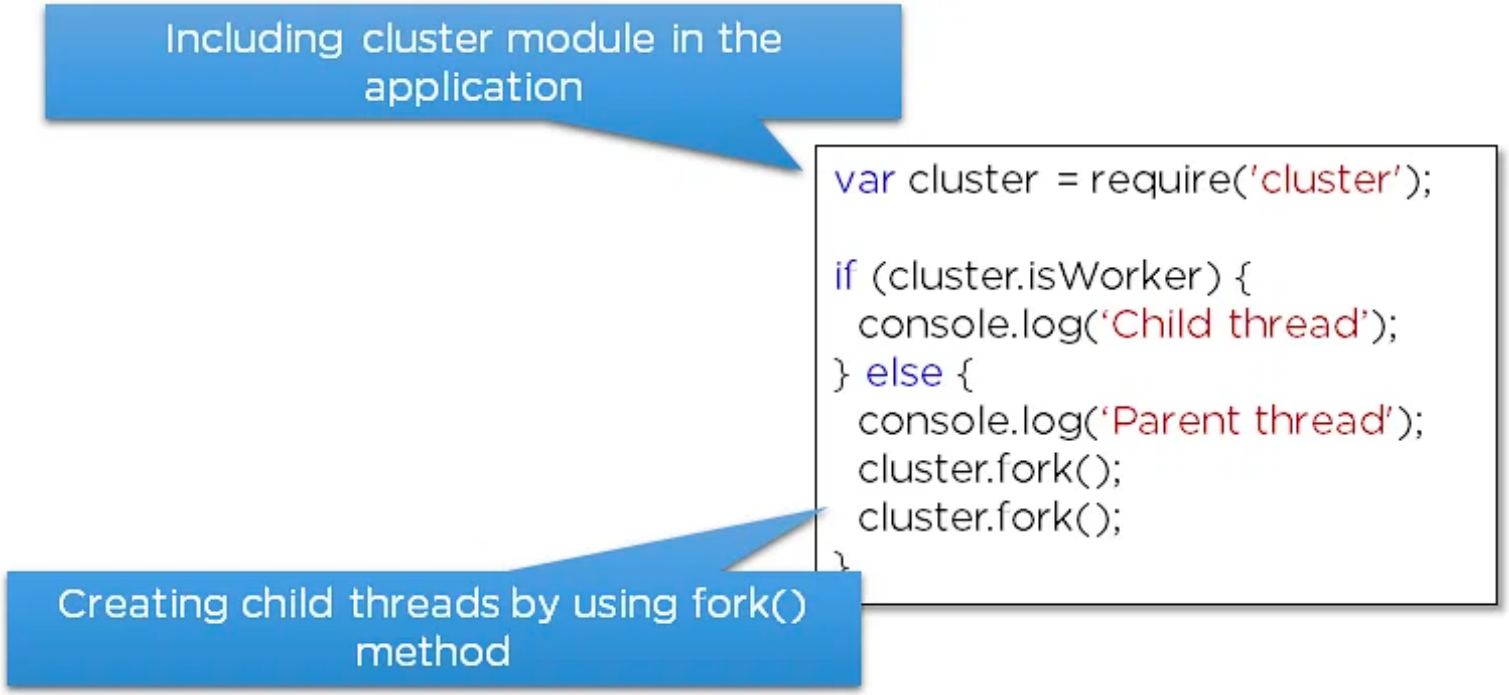
Nodejs components – global

Global objects in Node.js are available in all modules. These objects are functions, modules, strings, etc.

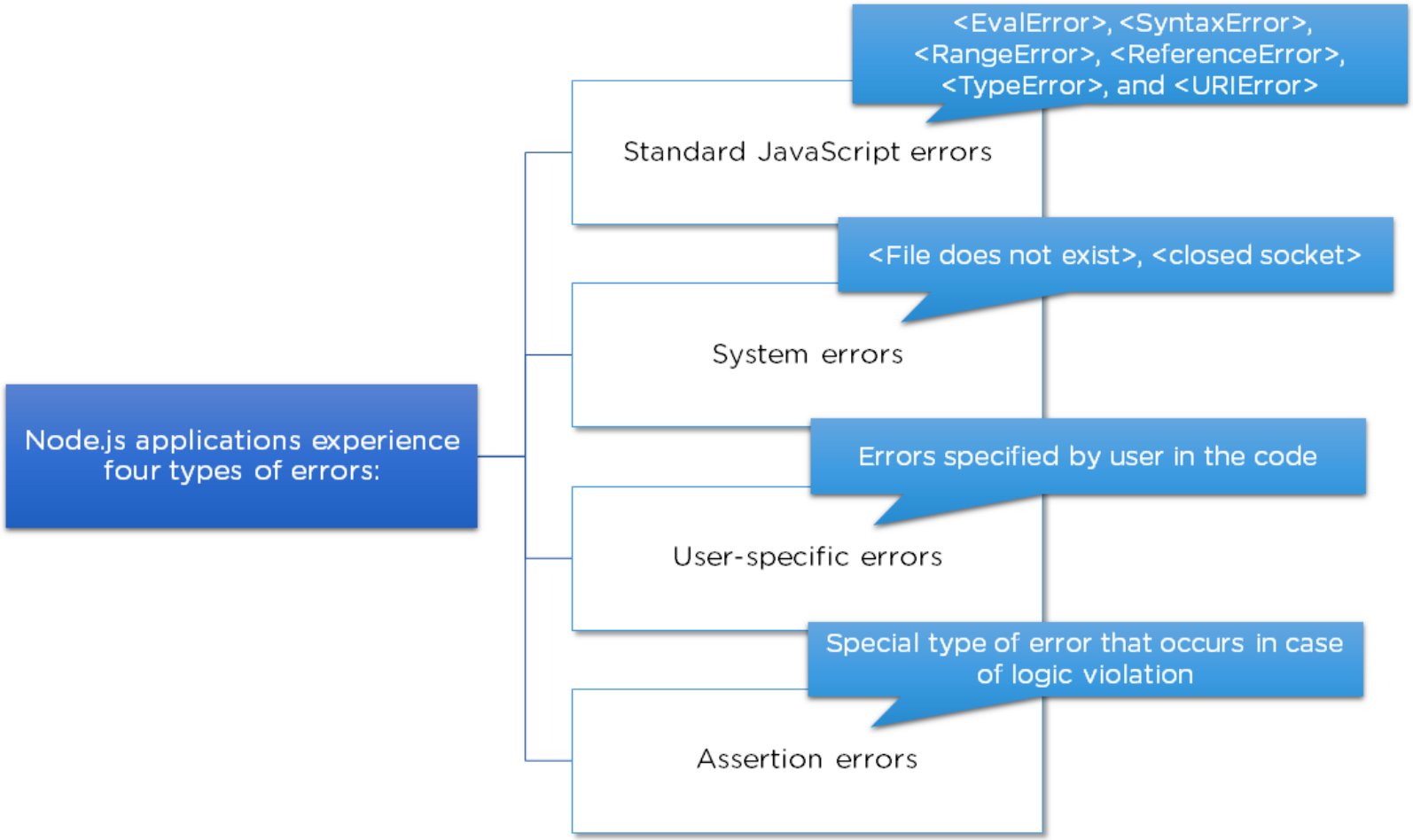
Global Objects	Description
__dirname	Specifies the name of the directory that contains the code of application
__filename	Specifies the filename of the code
exports	A reference to the module.exports, shorter to type
module	A reference to the current module
require	Used to import modules, local files, and also JSON

Nodejs components – cluster

Node.js is built-on on the concept of single-threaded programming. Cluster is a module that allows multi-threading by creating child processes that share the same server port and run simultaneously.



Node.js applications experience four types of errors.




```
try {
   var m = 1;
   var n = 1/0;
    }
catch (err) {
// Handling the error here.
}
```

Streams are the objects that let you read data or write data continuously. There are four types of streams:

1. **Readable**: These are the types of streams from which data can be read
2. **Writable**: These are the types of streams to which data can be written
3. **Duplex**: These are both readable and writable streams
4. **Transform**: Streams that can manipulate the data while it is being read or written

Buffer is a module that allows the handling of streams that contain only binary data. An empty buffer of length '10' can be created by this method:

```
var buf = Buffer.alloc(10);
```

The domain module intercepts errors that remain unhandled. Two methods are used for intercepting these errors:

1. **Internal Binding**: Error emitter executes its code inside the run method
2. **External Binding**: Error emitter is explicitly added to a domain via its add method

DNS module is used to connect to a DNS server and perform name resolution by using the following method:

```
dns.resolve()
```

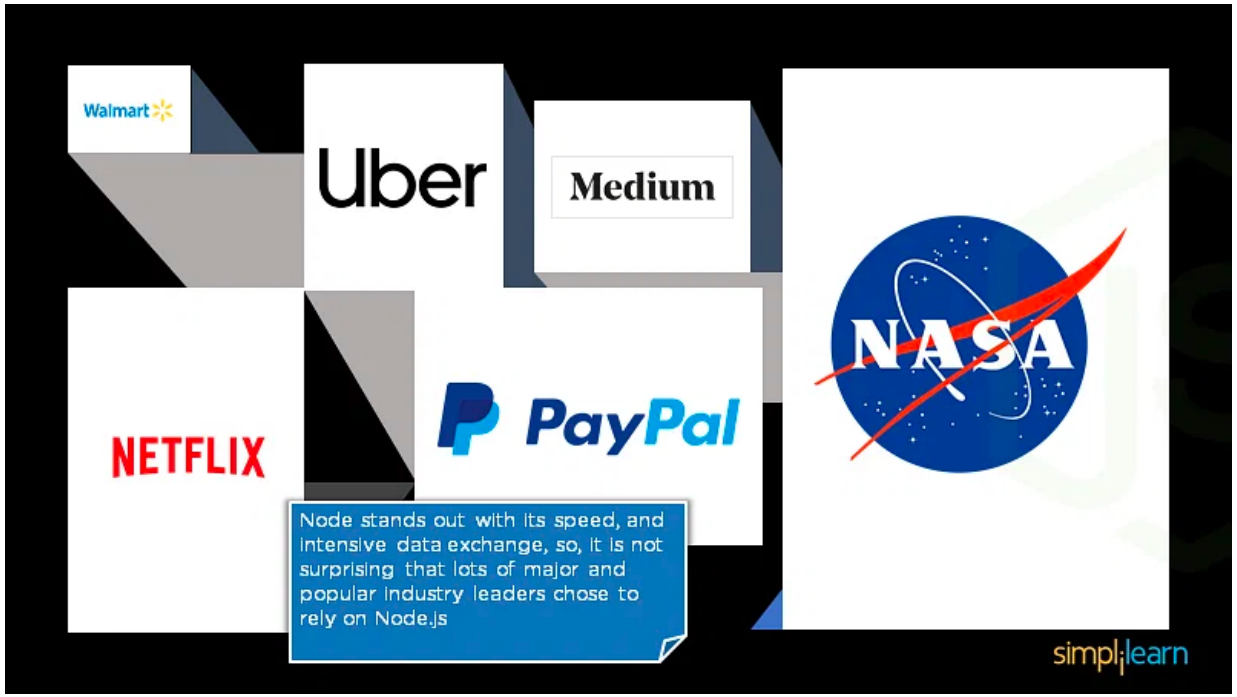
DNS module is also used for performing name resolution without a network communication by using the following method:

```
dns.lookup()
```

Node.js includes a debugging utility that can be accessed by a built-in debugging client. Node.js debugger is not feature-packed but supports the simple inspection of code. The debugger can be used in the terminal by using the 'inspect' keyword before the name of the JavaScript file. In order to inspect a file—myscript.js, for example—you can follow this method:

```
$ node inspect myscript.js
```

Nodejs components – customers



Netflix:
- Application scalability
- Data-intensive application

Walmart:
- Asynchronous I/O
- Efficient handling of concurrent requests

Uber:
- Asynchronous I/O
- Quick iterations
- Active open-source community

NASA:
- Reduced access times
- Ability to handle data-intensive tasks
- Capability to keep the server active 24/7

Paypal:
- Extremely fast build times
- Fewer lines of code
- Ability to handle large amounts of data

Medium:
- Data-driven applications
- Ability to run A/B tests
- Simple server maintenance

Nodejs – when not to use

- NodeJS is a very CPU heavy application , when it comes to heavy computation, NodeJS is not the best option in hand. Other than NodeJs, there are plenty of better solutions available for CPU intensive applications.

- When using NodeJs with basic crud or HTML application there is no need for high hopes . Though NodeJs will make it more scalable, there is no need for expecting a traffic flood only because the application is powered by NodeJs. So when the data is provided straightforwardly by, by the server and specially where there is no need to have a separate API, in this case there is no need for using NodeJs.

- For Relational database access type of projects, using NodeJs is not an efficient idea at all. In comparison with the other framework's tool boxes, the relational database tools of NodeJs are not that robust, reliable and easy to work with.

Thank you