Node.js for Java Engineers

Ryan Stevens

December 5, 2014

About me

- @ryan_stevens
- Principal Consultant @nodesource
- Have programmed for / managed
  - JavaScript - frontend teams
  - Node.js teams
  - Java teams


PUJOLS
5

Goals

- Create a valid "this for that" model
- Understand key differences, similarities and tradeoffs
- Breakdown of Node.js Java developers can reason through
- Maybe, just maybe…

  Node.js developers could stand to learn a little about **Java**

This Talk Isn't…

- A syntax / language comparisons
- A comparison of the "best of Java" vs "best of Node.js"

Frontend Backend

Spring MVC

Controllers

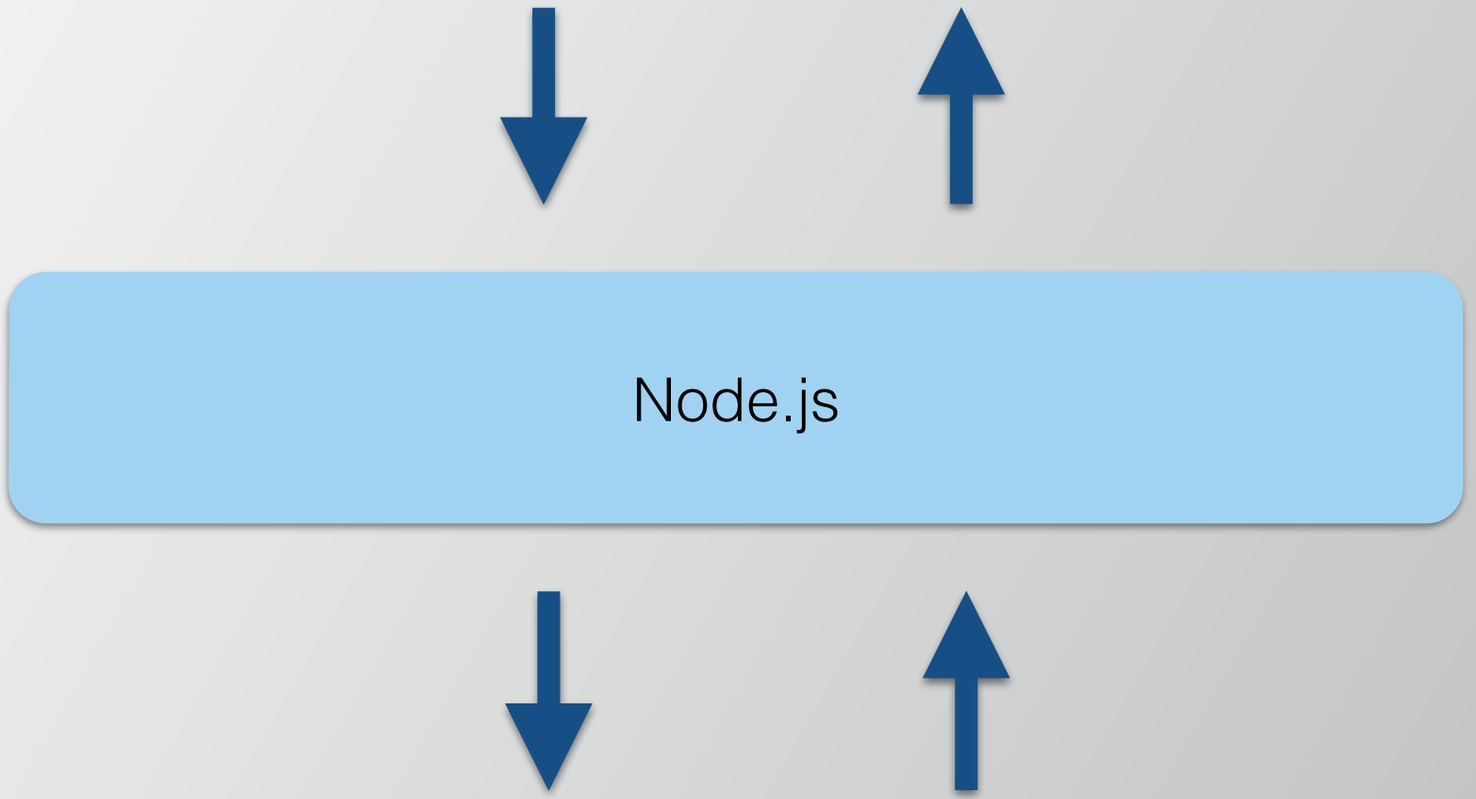
Views

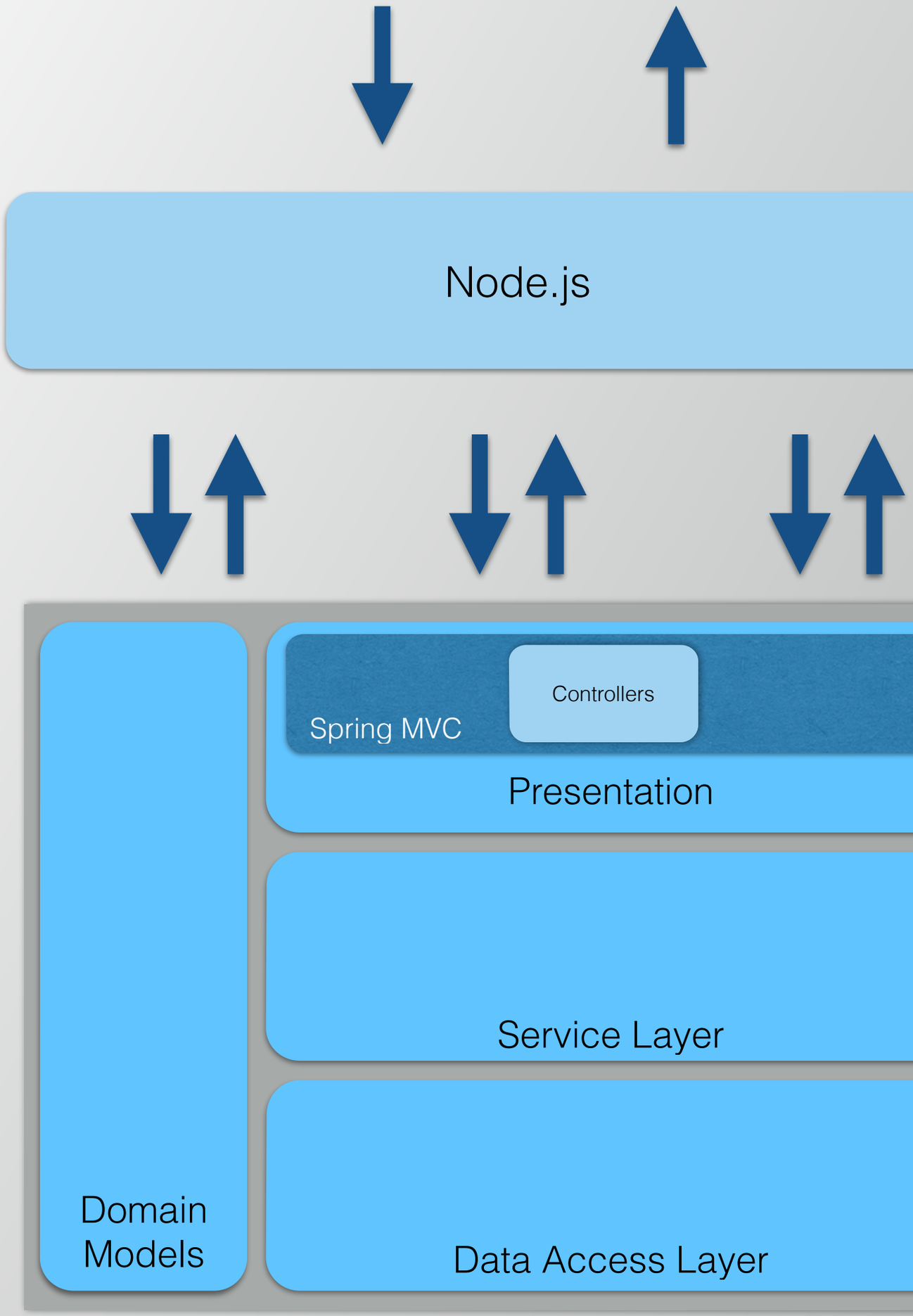
Presentation

Service Layer

Domain Models

Data Access Layer


Node.js
Spring MVC
Controllers
Presentation
Service Layer
Data Access Layer
Domain
Models


Node.js
Spring MVC
Controllers
Presentation
Service Layer
Data Access Layer
Domain
Models

Node As Your Frontend Backend

Strengths

- Decouples release cycles across projects
- Node can focus on product problems
- Java backends can focus on data problems

Tradeoffs

- Adds latency
- Deploy complexity between model changes
- Harder to go “full stack”

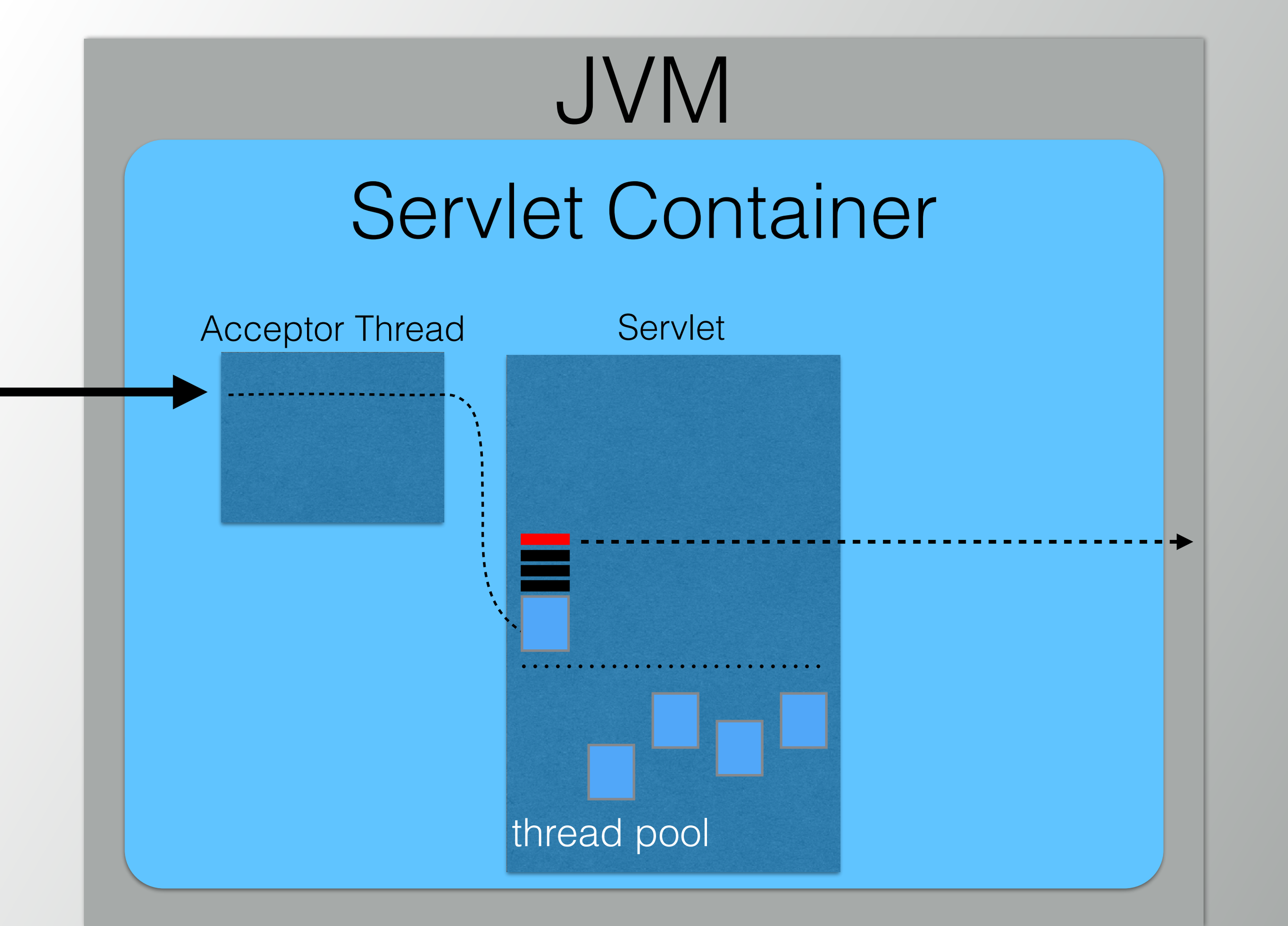
Common Building Blocks

HTTP Req / Res Lifecycle

Node		**Java**
• Built in HTTP module	**is lower level than**	• Servlet Containers
• Connect / Express	**is a subset of**	• Spring MVC
• Adding {route : handlers} to a framework	**isn't as magic as**	• Servlets init and register with container
• Single threaded, event loop	**Is a completely different paradigm than**	• Multi-threaded, thread pools
• Middleware	**provides a mechanism similar to**	• Filters, Interceptors

Basic Servlet Code

```
@WebServlet(urlPatterns = “/")
public class Length extends HttpServlet {

  @Override
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException {
    int length = getResponseLength("https://nodesource.com/");
    response.getWriter().write("Length " + length);

  }

  private int getResponseLength(String url)  {
    //do a sync HTTP get
    return html.length();
  }
}
```

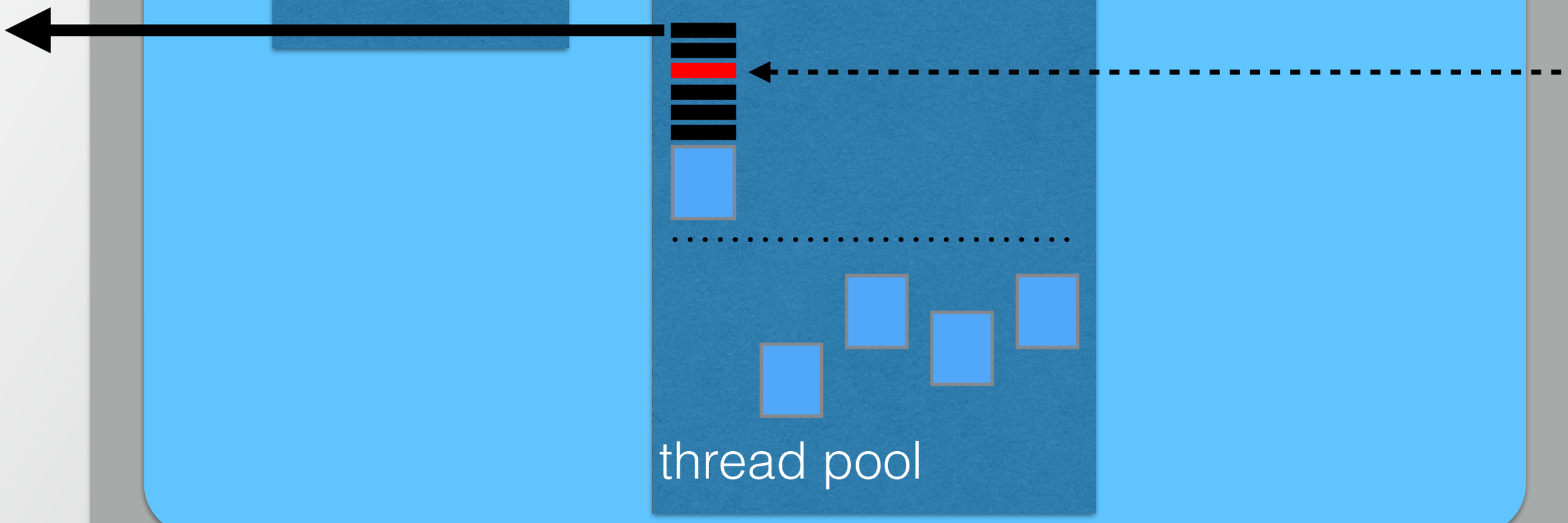

JVM
Servlet Container
Acceptor Thread
Servlet
thread pool

JVM

Servlet Container

Acceptor Thread

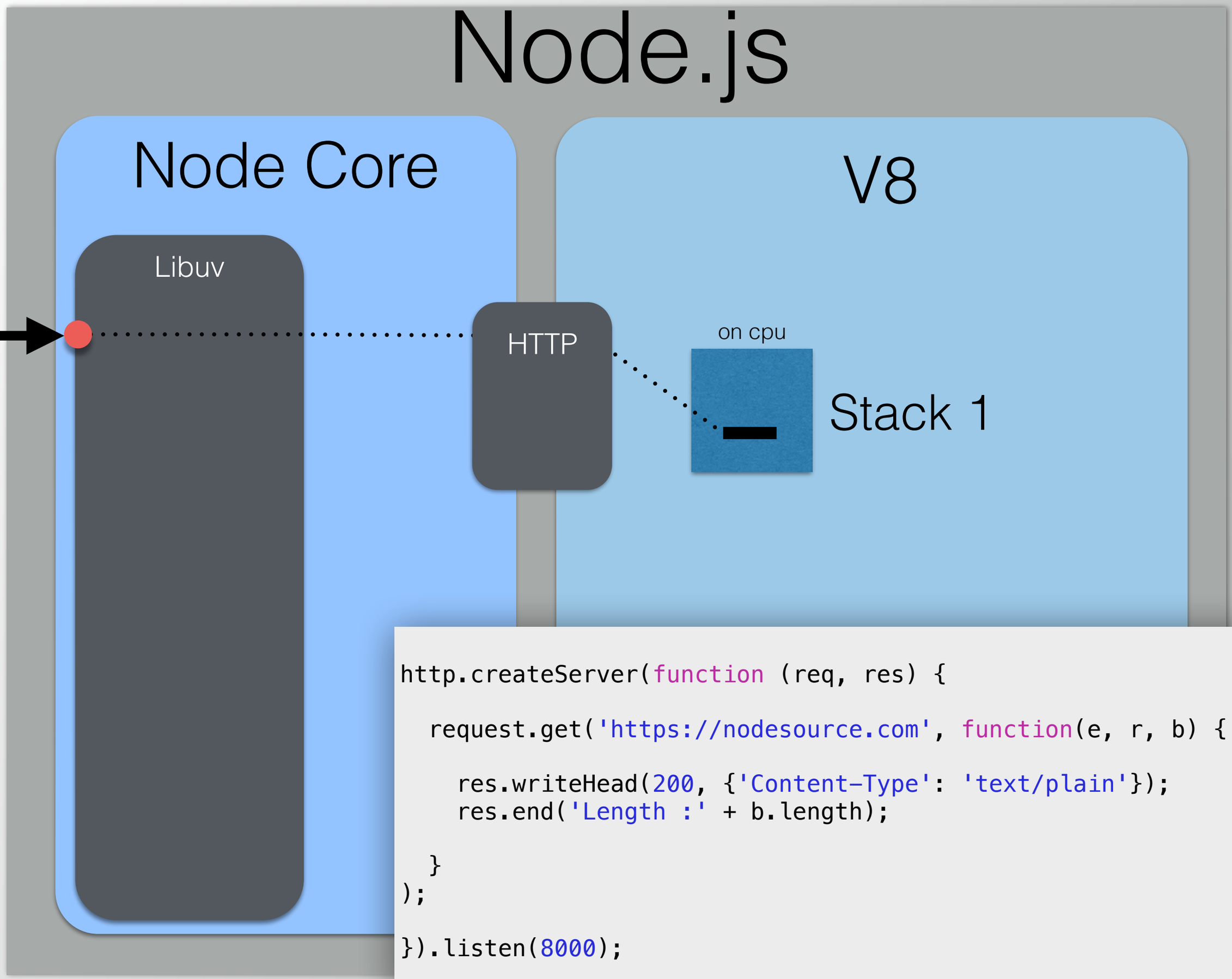
Servlet



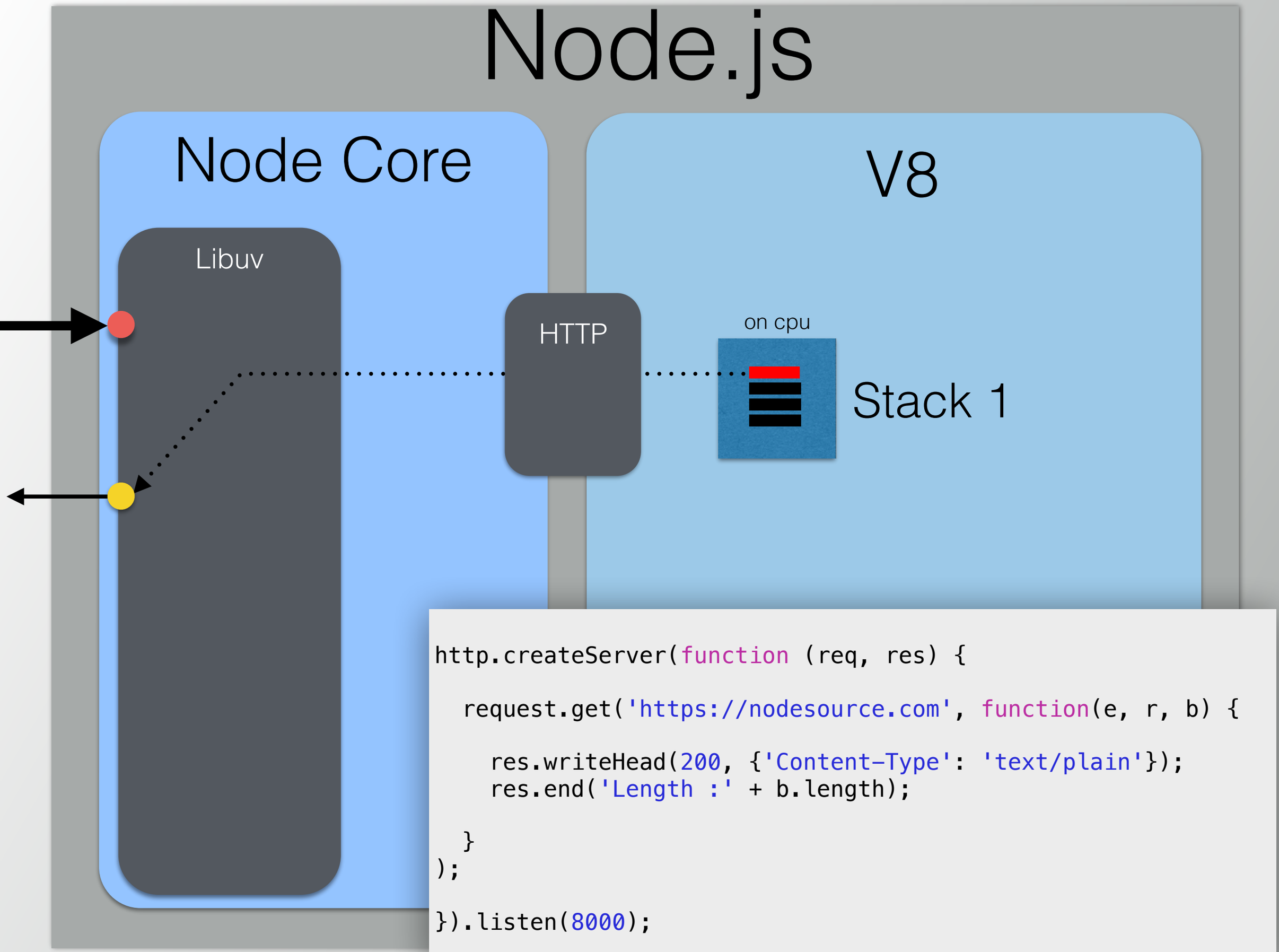
thread pool

Basic Node

```
var http    = require('http'),
    request = require('request');

http.createServer(function (req, res) {
  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  });

}).listen(8000);
```




```
http.createServer(function (req, res) {

  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  }
);

}).listen(8000);
```




```
http.createServer(function (req, res) {

  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  }
);

}).listen(8000);
```

Node.js

Node Core

Libuv

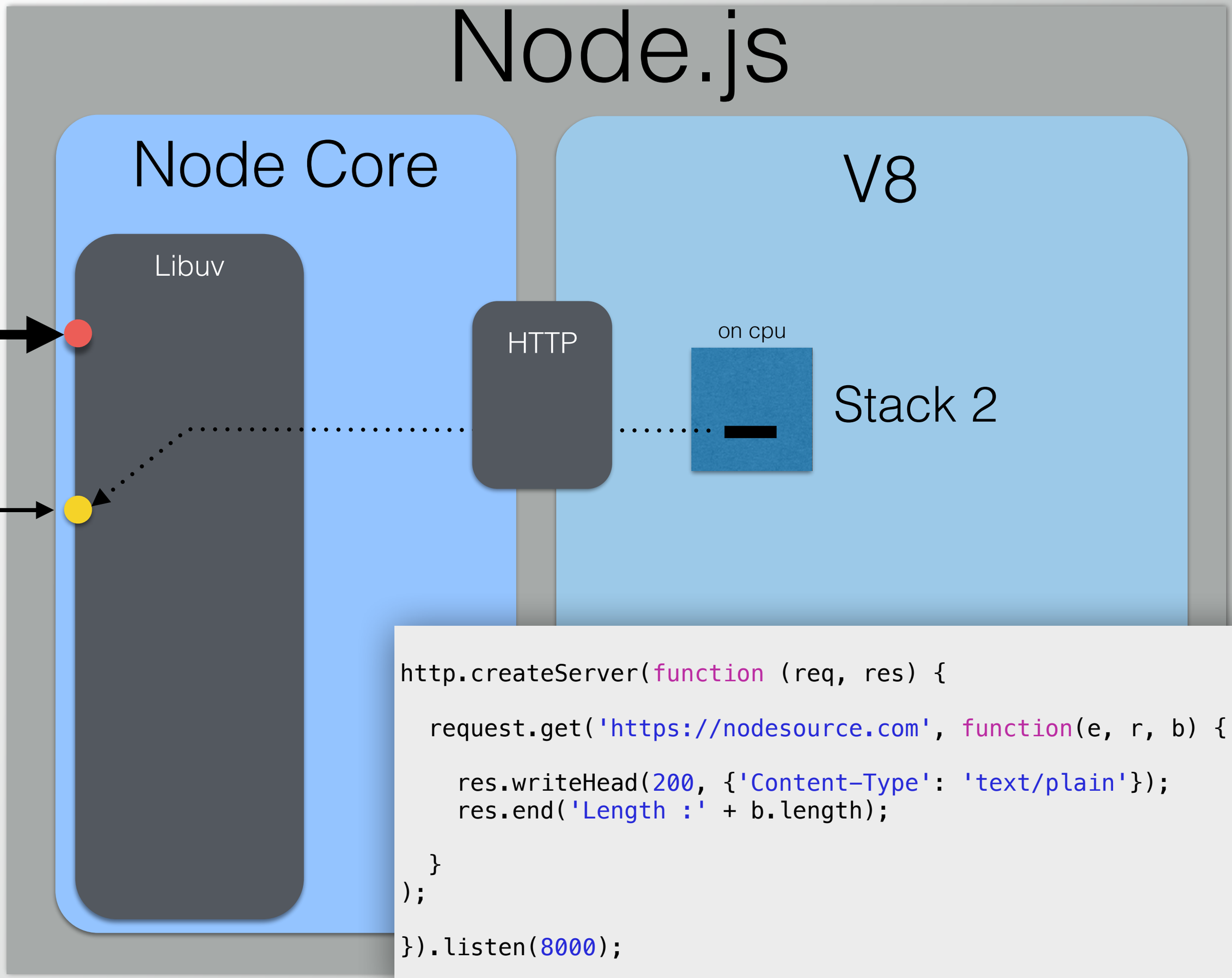
HTTP

V8

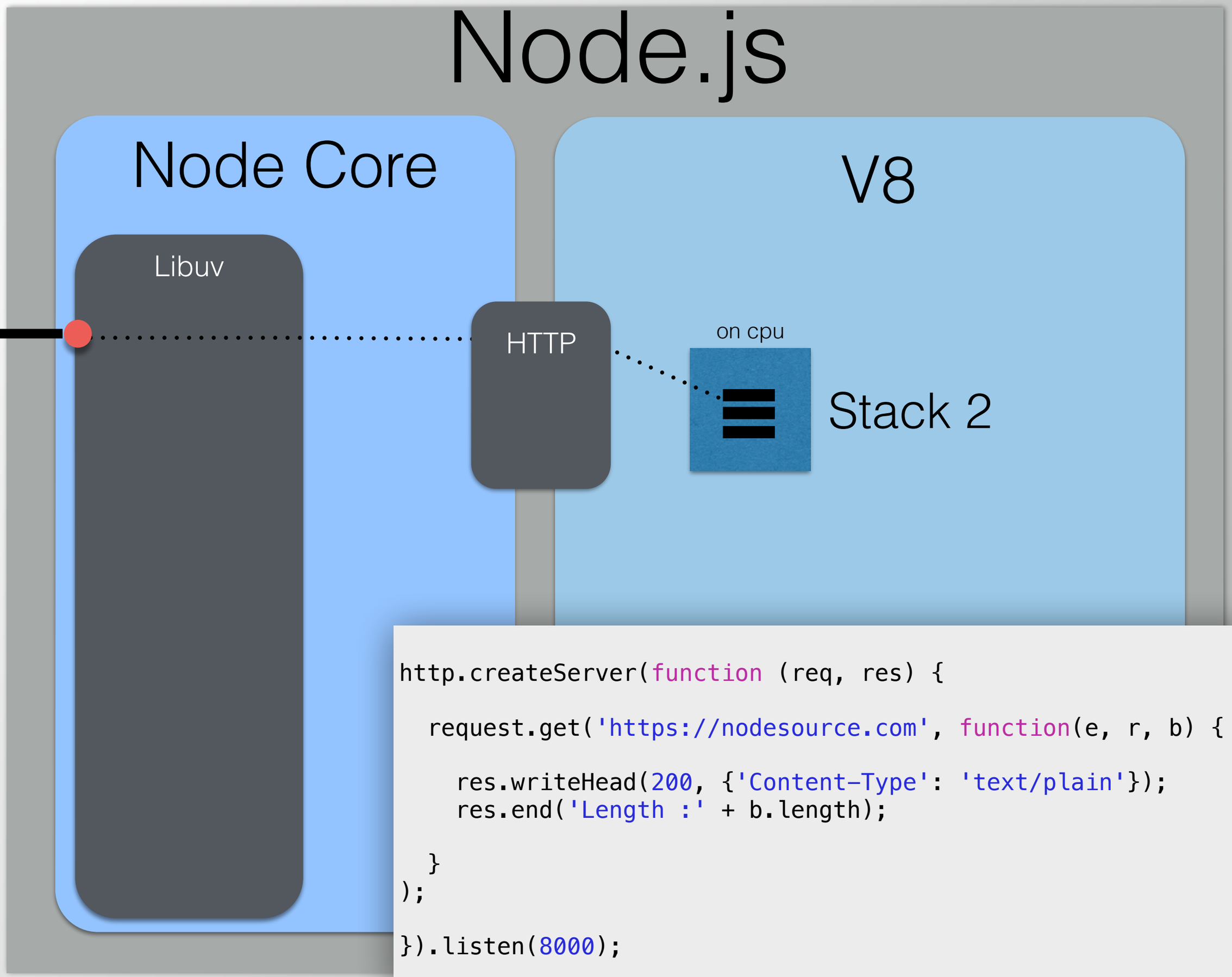
on cpu

Stack 1 done

```
http.createServer(function (req, res) {

  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  }
);

}).listen(8000);
```




```
http.createServer(function (req, res) {

  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  }
);

}).listen(8000);
```




```
http.createServer(function (req, res) {

  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  }
);

}).listen(8000);
```

Key Differences

- Node http module is very low level
  - Users program explicitly has to create a server
  - No built-in ability to partition URL routes
- Servlet containers gives you much more out the box
  - A quick way to plug in more servlets
  - Mechanisms for reading config and context

Key Similarities

Java

- HttpServletRequest
  - Contains a readable stream
- HttpServletResponse
  - getOutputStream().write(buffer);
  - getWriter().write(“hello”);

Node

- http.IncomingMessage
  - Is a readable stream
- http.ServerResponse
  - write(buffer, ‘binary’);
  - write(‘hello’);
  - end();

Express with Node

```
var app = require('express')(),
    request = require(‘request’),
    uppercase = require(‘toupper-middleware’);

app.get('/', function (req, res) {
   var url = ‘https://nodesource.com';
   request.get(url, function(err, resp, body) {
     res.writeHead(200, {'Content-Type': 'text/plain'});
     res.end('Length :' + body.length);
   });

});

app.use(uppercase());
var server = app.listen(3000);
```

Java Filter

```
public void doFilter(ServletRequest request,
                     ServletResponse response,
                     FilterChain chain)
                     throws IOException, ServletException {
    ServletResponse newResponse =
      new CharResponseWrapper((HttpServletResponse) response);

    chain.doFilter(request, newResponse);

    response.getWriter().write(newResponse.toString());

}
```

Java Filter

```
class CharResponseWrapper extends HttpServletResponseWrapper {
    protected CharArrayWriter charWriter;
    protected PrintWriter writer;

    public CharResponseWrapper(HttpServletResponse response) {
      super(response);
      charWriter = new CharArrayWriter();
    }

    @Override
    public PrintWriter getWriter() throws IOException {
      if (writer != null) return writer;
      writer = new PrintWriter(charWriter);
      return writer;
    }

    public String toString() {
      String s = charWriter.toString();
      return s.toUpperCase();
    }
}
```

Node Middleware

```
module.exports = function create() {
  return function toUpper(req, res, next) {

    var oldWrite = res.write;

    res.write = function toUpperWrite(str) {
      if (!str) return;
      oldWrite.call(res, str.toUpperCase());
    };

    next();
  };
};
```

Streams

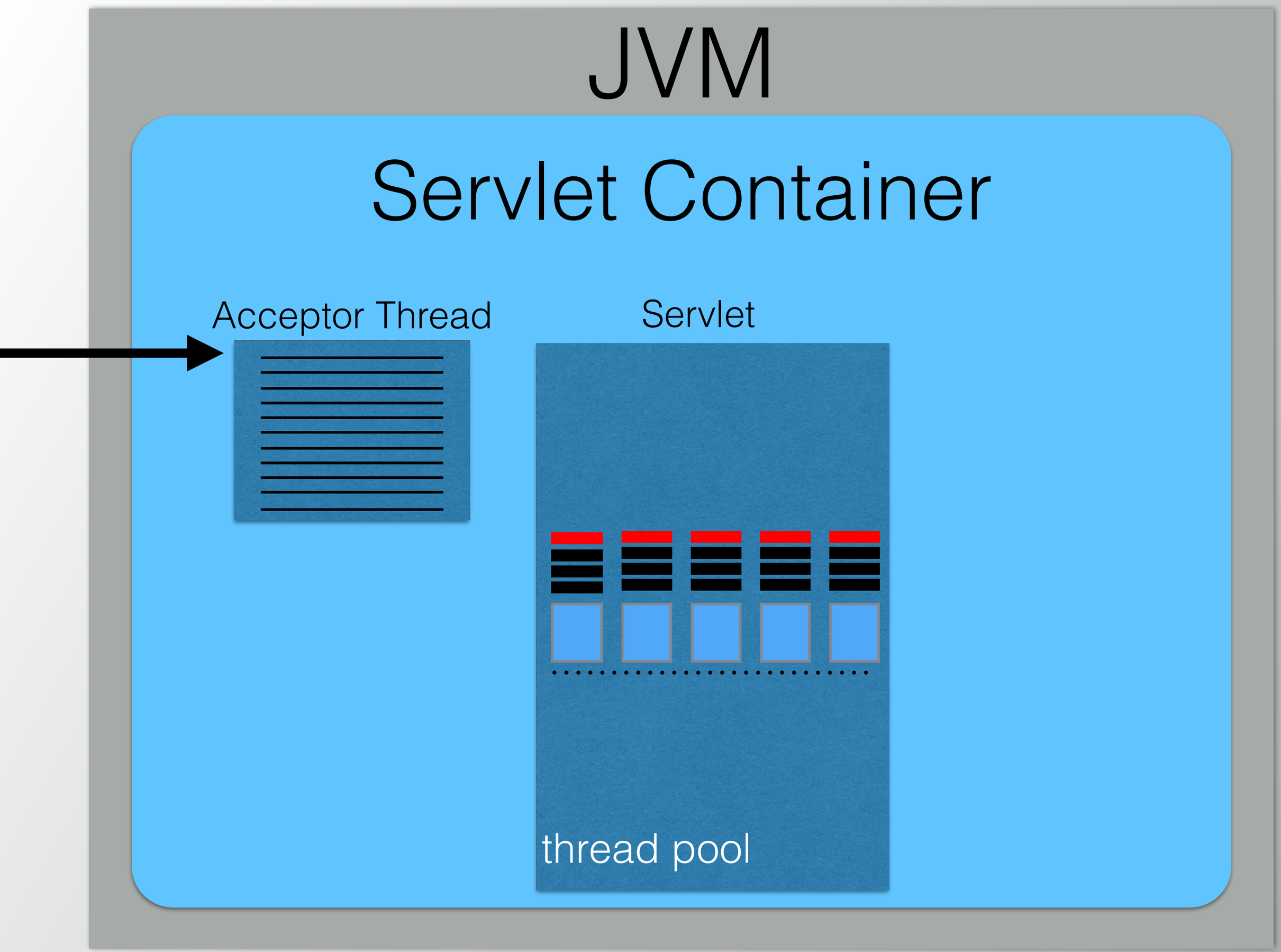
- Have been part of both communities / ecosystem from the beginning

- In Java, many programmers consciously avoid working directly with them

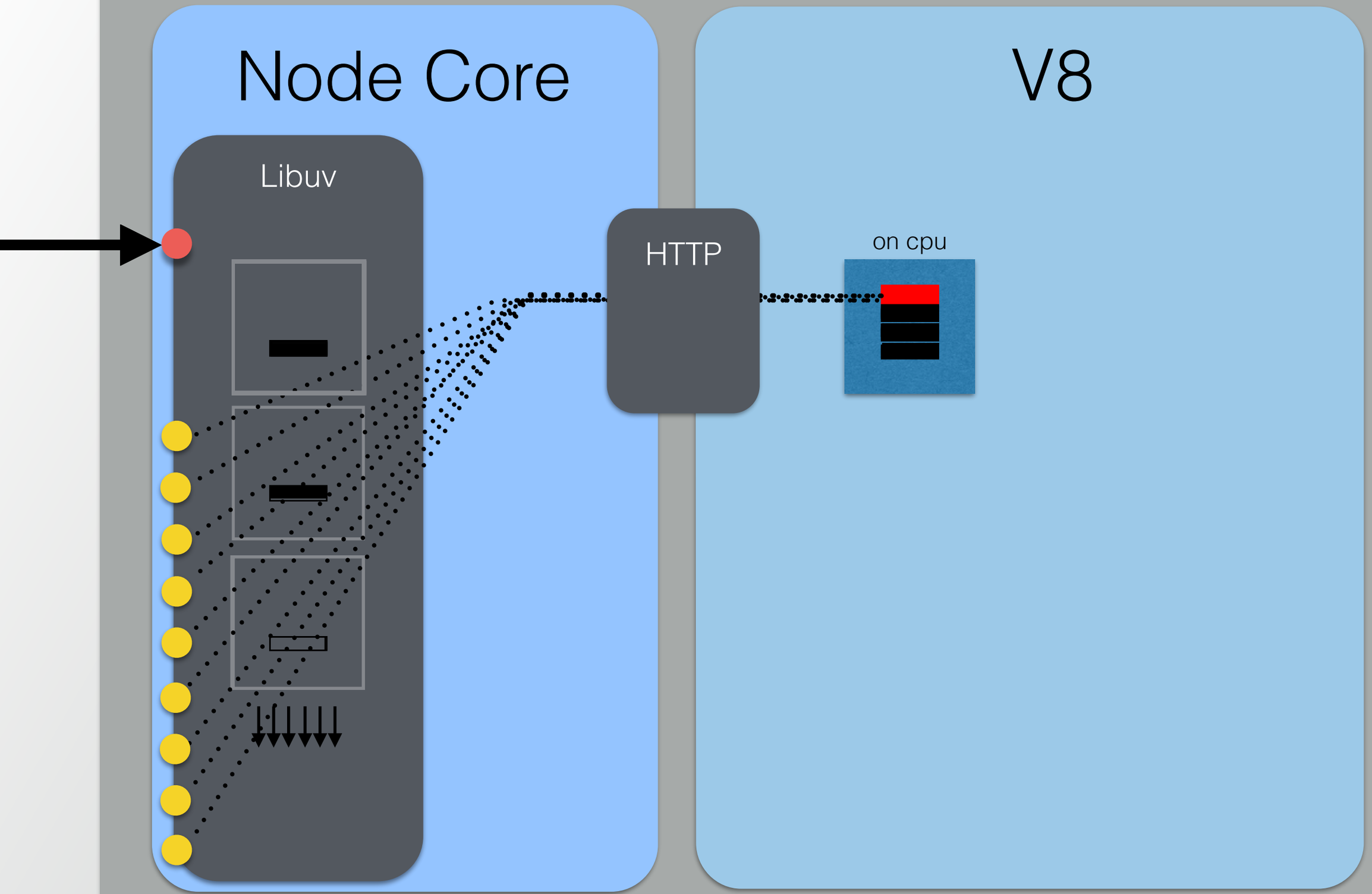
- In Node, streams are **fun** to work with

Let’s talk about threads


THERE CAN BE ONLY ONE


JVM
Servlet Container
Acceptor Thread
Servlet
thread pool

Node.js



Java's NIO is like Node.js

Ja NIO i Node.js

Kinda

BUT NOT REALLY

Java's NIO is more like Libuv

Java's N… like Libuv

NOT COMPLETELY WRONG

Servlet Code - Spring / Async

```
@RequestMapping("/length")
public @ResponseBody Callable<String> length() {

  return new Callable<String>() {
    @Override
    public String call() throws Exception {
      return Integer.toString(
       getResponseLength(“https://nodesource.com")
      );
    }
  };

}

public int getResponseLength(String url) {
  //make an HTTP synchronous call
}
```

Servlet Code - Spring / Async

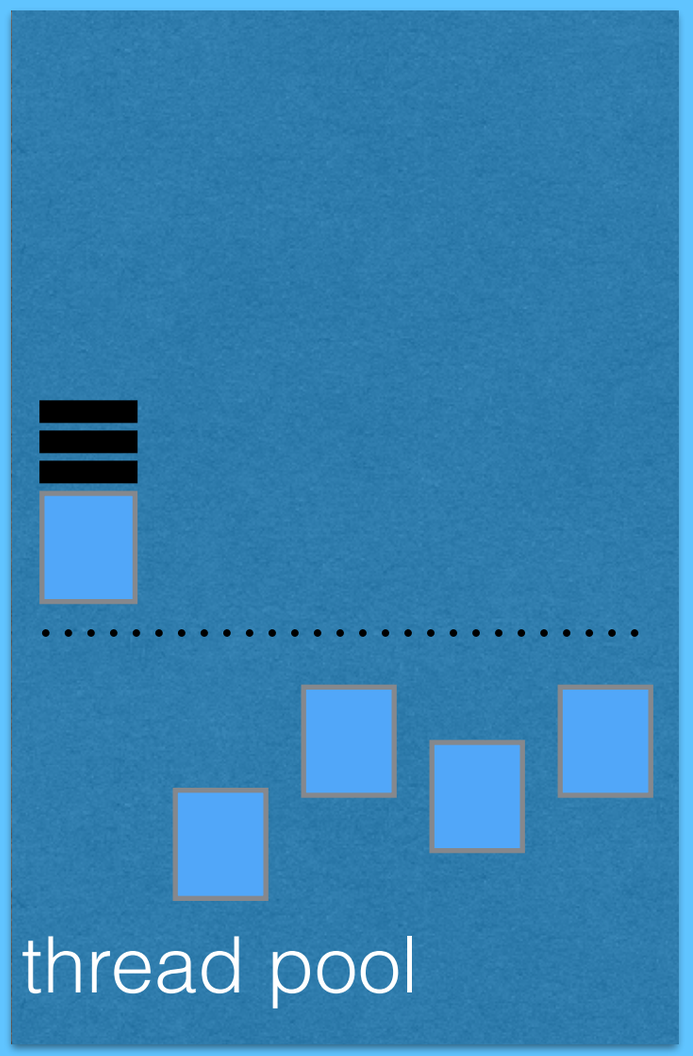
```
@RequestMapping("/length")
public @ResponseBody Callable<String> length() {

  return new Callable<String>() {
    @Override
    public String call() throws Exception {
      return Integer.toString(
       getResponseLength(“https://nodesource.com")
      );
    }
  };

}

public int getResponseLength(String url) {
  //make an HTTP synchronous call
}
```

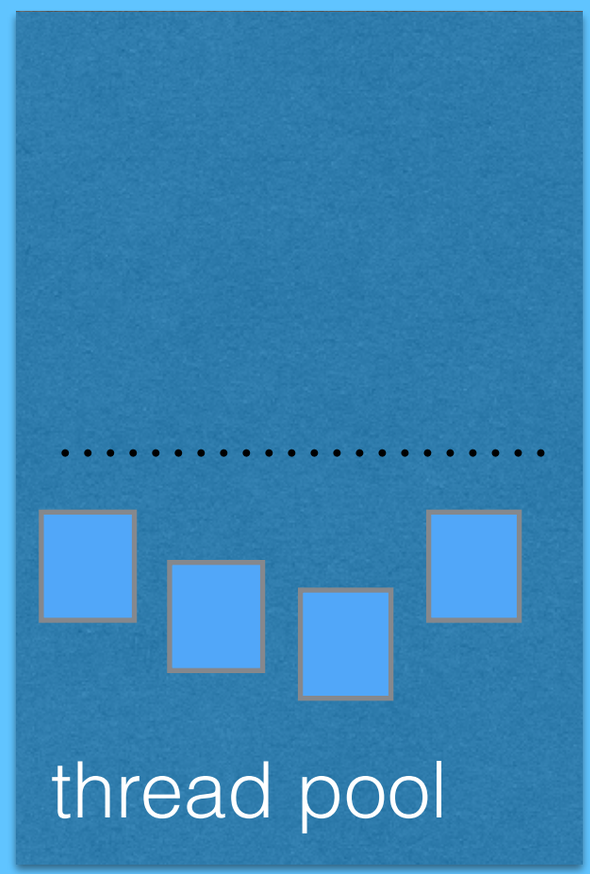
JVM

Servlet Container

Acceptor Thread

Servlet





JVM

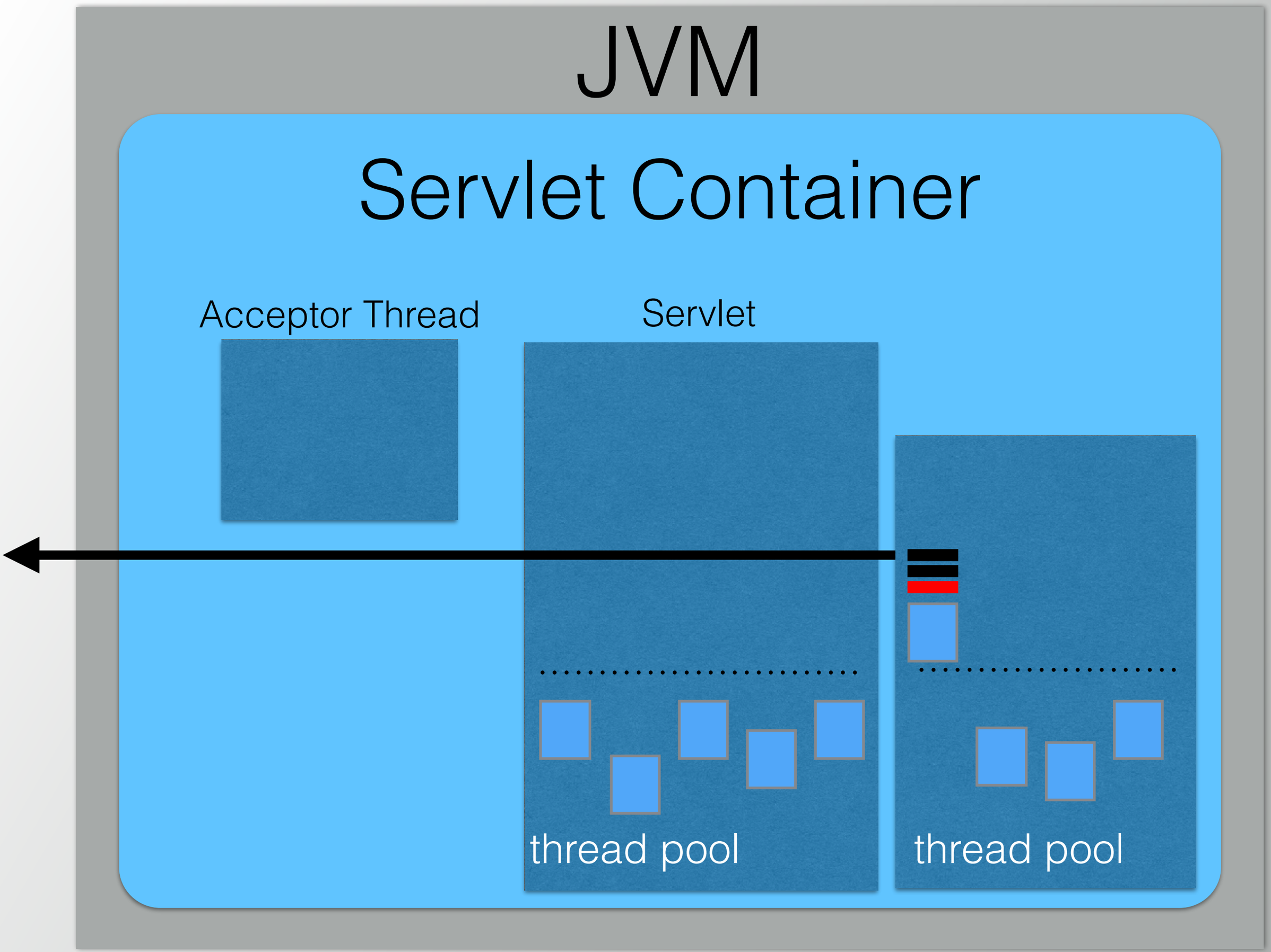
Servlet Container

Acceptor Thread

Servlet

thread pool

thread pool


JVM
Servlet Container
Acceptor Thread
Servlet
thread pool
thread pool

Servlet Code - Spring / Async

```
@RequestMapping("/lengthAsync")
public @ResponseBody DeferredResult<String> length() throws IOException {

   final DeferredResult<String> result = new DeferredResult();
   AsyncHttpClient client = new AsyncHttpClient();

   client.prepareGet(“http://nodesource.com")
     .execute(new AsyncCompletionHandlerBase() {

       @Override
       public Response onCompleted(Response response) throws Exception {
         int len = response.getResponseBody().length();
         result.setResult(Integer.toString(len));
         return super.onCompleted(response);
       }
     }
   );

   return result;
}
```

Servlet Code - Spring / Async

```
@RequestMapping("/lengthAsync")
public @ResponseBody DeferredResult<String> length() throws IOException {

   final DeferredResult<String> result = new DeferredResult();
   AsyncHttpClient client = new AsyncHttpClient();
   client.prepareGet(“http://nodesource.com")
     .execute(new AsyncCompletionHandlerBase() {

       @Override
       public Response onCompleted(Response response) throws Exception {
         int len = response.getResponseBody().length();
         result.setResult(Integer.toString(len));
         return super.onCompleted(response);
       }
     }
   );

   return result;
}
```

JVM

Servlet Container

Acceptor Thread

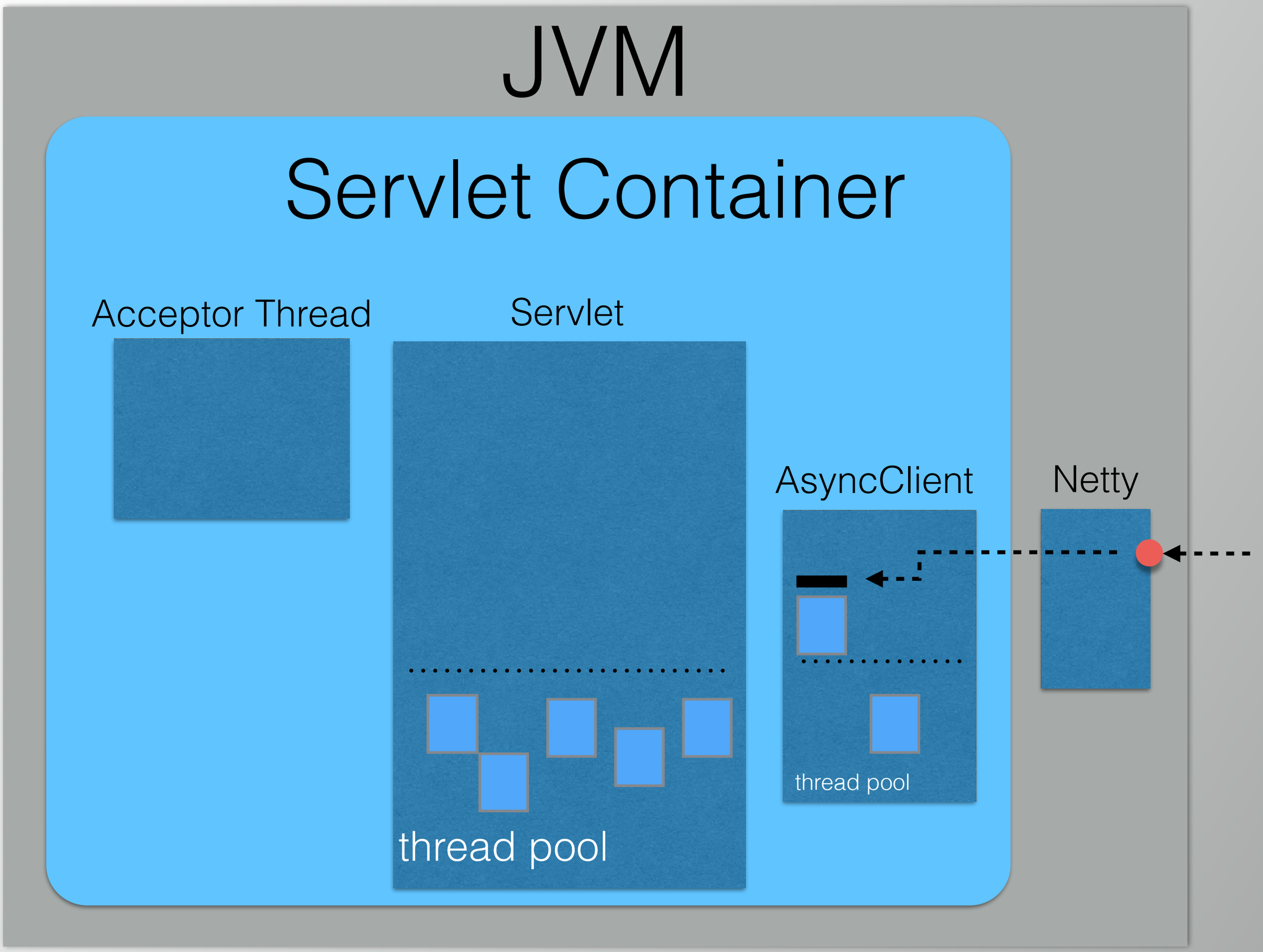
Servlet

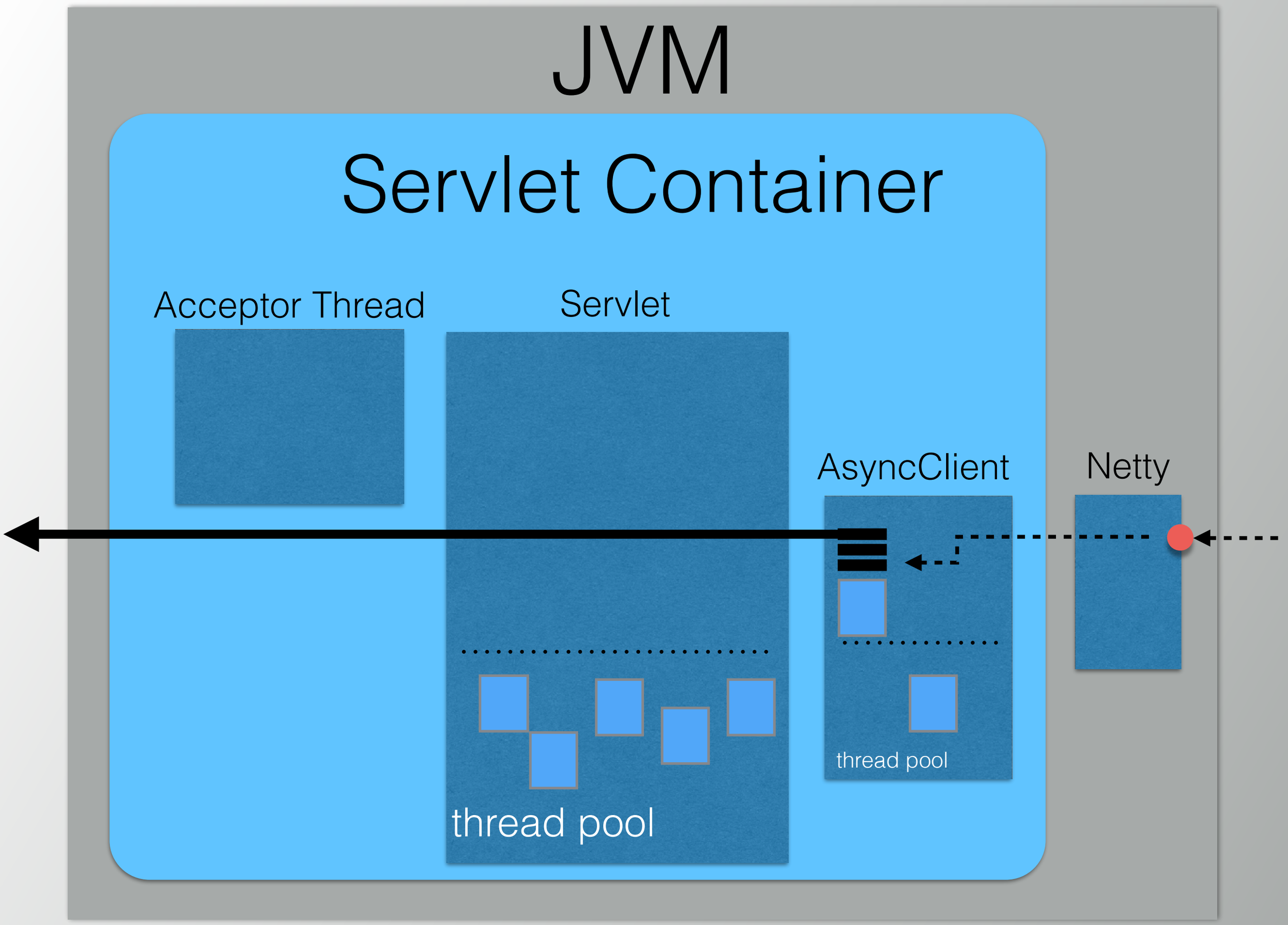
AsyncClient

Netty

thread pool

thread pool


JVM
Servlet Container
Acceptor Thread
Servlet
thread pool
AsyncClient
thread pool
Netty


JVM
Servlet Container
Acceptor Thread
Servlet
AsyncClient
Netty
thread pool
thread pool

JVM Thread Pools

- Increases concurrency complexity
- Can share heap
- Multiple pools can compete for CPU
- Context switching is expensive

Node Event Loop

- Favors simplicity over vertical scaling
- One heap per process
- Extremely “fair” and deterministic
- Invoking a closure (callback) is very fast

Netty is like Libuv

NIO's Slow Adoption

- Thread pools have provided a practical balance
- Heavily reliant upon servlet containers implementation to be useful
- Combining asynchronous programming makes concurrent programming even harder
- Castles have been built on old foundations

Asynchronous, event loop based programming is a paradigm shift

Asynchronous, event loop based programming is a paradigm shift

In Node, programmers approach problems async first

Try Catch & Call Stacks

Servlet Async

```
@RequestMapping("/lengthAsync")
public @ResponseBody DeferredResult<String> length() throws IOException {

   AsyncHttpClient client = new AsyncHttpClient();
   final DeferredResult<String> result = new DeferredResult();

   client
     .prepareGet(“http://nodesource.com")
     .execute(new AsyncCompletionHandlerBase() {

       @Override
       public Response onCompleted(Response response) throws Exception {
         int len = response.getResponseBody().length();
         result.setResult(Integer.toString(len));
         return super.onCompleted(response);
       }
     }
   );

   return result;
}
```

Servlet Async

```
@RequestMapping("/lengthAsync")
public @ResponseBody DeferredResult<String> length() throws Exception {
   if (10>0) throw new Exception("ERROR");
   AsyncHttpClient client = new AsyncHttpClient();
   final DeferredResult<String> result = new DeferredResult();

   client
     .prepareGet("http://nodesource.com")
     .execute(new AsyncCompletionHandlerBase() {

       @Override
       public Response onCompleted(Response response) throws Exception {
         int len = response.getResponseBody().length();
         result.setResult(Integer.toString(len));
         return super.onCompleted(response);
       }
     }
   );

   return result;
}
```

Servlet stack trace

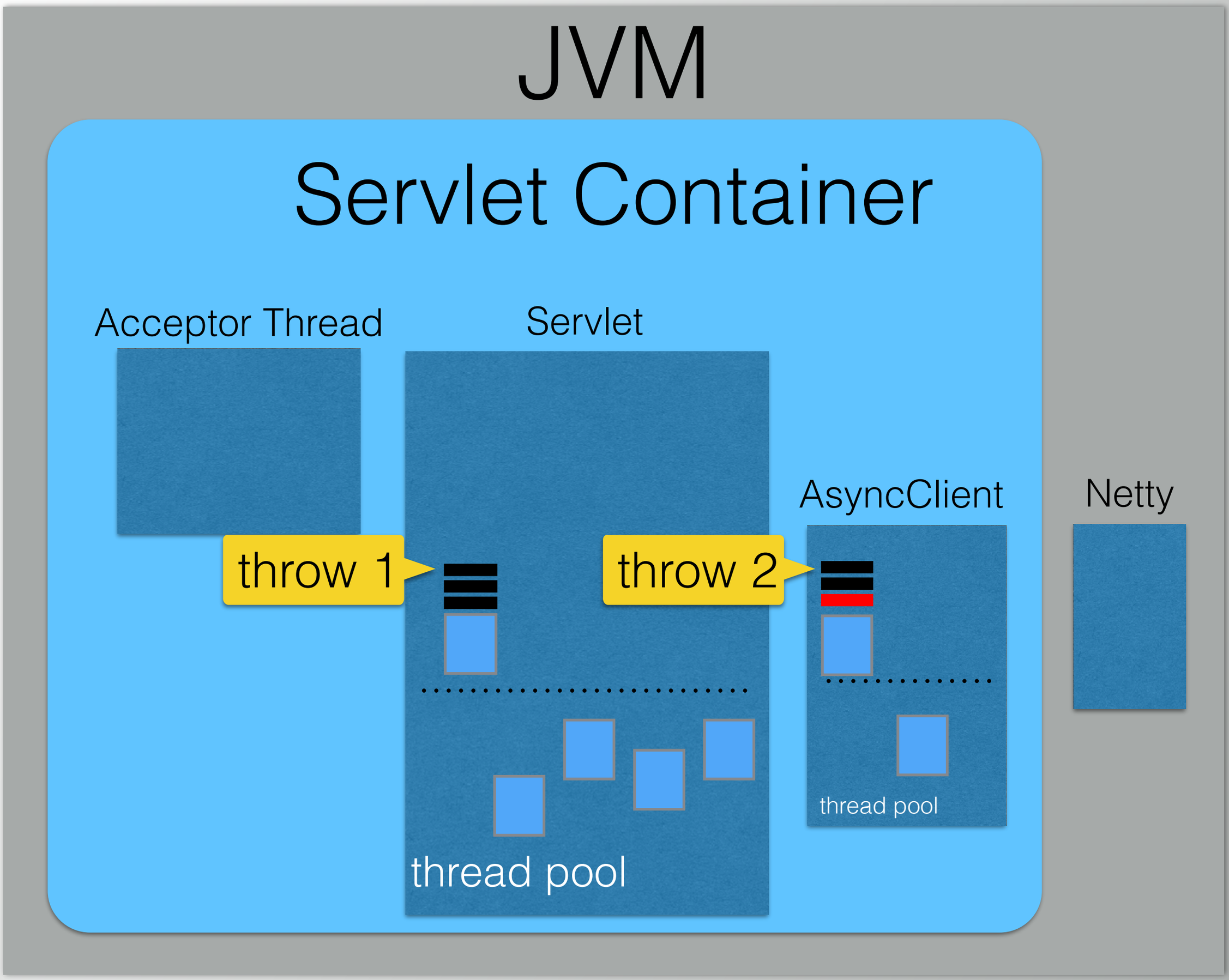
```
at org.springframework.samples.mvc.async.CallableController.length(CallableController.java:102)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:606)
at org.springframework.web.method.support.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java:215)
at org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:132)
at org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:104)
at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandleMethod(RequestMappingHandlerAdapter.java:781)
at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:721)
at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:83)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:943)
at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:877)
at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:961)
at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:852)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:837)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:305)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
at org.springframework.security.web.csrf.CsrfFilter.doFilterInternal(CsrfFilter.java:85)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107)
at org.springframework.web.filter.DelegatingFilterProxy.invokeDelegate(DelegatingFilterProxy.java:344)
at org.springframework.web.filter.DelegatingFilterProxy.doFilter(DelegatingFilterProxy.java:261)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:222)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:123)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:502)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:171)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:100)
at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:953)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:118)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:408)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1041)
at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:603)
at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:310)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
at java.lang.Thread.run(Thread.java:745)
```

Servlet Async

```
@RequestMapping("/lengthAsync")
public @ResponseBody DeferredResult<String> length() throws Exception {

   AsyncHttpClient client = new AsyncHttpClient();
   final DeferredResult<String> result = new DeferredResult();

   client
     .prepareGet(“http://nodesource.com")
     .execute(new AsyncCompletionHandlerBase() {

       @Override
       public Response onCompleted(Response response) throws Exception {
         if (10>0) throw new Exception(“ERROR");
         int len = response.getResponseBody().length();
         result.setResult(Integer.toString(len));
         return super.onCompleted(response);
       }
     }
   );

   return result;
}
```

AsyncClient stack trace

```
at org.springframework.samples.mvc.async.CallableController$2.onCompleted(CallableController.java:128)
at com.ning.http.client.AsyncCompletionHandlerBase.onCompleted(AsyncCompletionHandlerBase.java:25)
at com.ning.http.client.AsyncCompletionHandler.onCompleted(AsyncCompletionHandler.java:63)
at com.ning.http.client.providers.netty.NettyResponseFuture.getContent(NettyResponseFuture.java:293)
at com.ning.http.client.providers.netty.NettyResponseFuture.done(NettyResponseFuture.java:320)
at com.ning.http.client.providers.netty.NettyAsyncHttpProvider.markAsDone(NettyAsyncHttpProvider.java:1532)
at com.ning.http.client.providers.netty.NettyAsyncHttpProvider.finishUpdate(NettyAsyncHttpProvider.java:1548)
at com.ning.http.client.providers.netty.NettyAsyncHttpProvider.access$2700(NettyAsyncHttpProvider.java:161)
at com.ning.http.client.providers.netty.NettyAsyncHttpProvider$HttpProtocol.handle(NettyAsyncHttpProvider.java:2239)
at com.ning.http.client.providers.netty.NettyAsyncHttpProvider.messageReceived(NettyAsyncHttpProvider.java:1227)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
at java.lang.Thread.run(Thread.java:745)
```


JVM
Servlet Container
Acceptor Thread
Servlet
AsyncClient
Netty
throw 1
throw 2
thread pool
thread pool

Node

```
var http   = require('http'),
    request = require('request');

http.createServer(function (req, res) {
  request.get('https://nodesource.com', function(e, r, b) {

    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  });

}).listen(8000, '127.0.0.1');
```

Node

```
var http   = require('http'),
    request = require('request');

http.createServer(function (req, res) {
  throw new Error('Error');
    request.get('https://nodesource.com', function(e, r, b) {

      res.writeHead(200, {'Content-Type': 'text/plain'});
      res.end('Length :' + b.length);

    });

}).listen(8000, '127.0.0.1');
```

Node

```
var http   = require('http'),
    request = require('request');

http.createServer(function (req, res) {
  request.get('https://nodesource.com', function(e, r, b) {
    throw new Error('Error');
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Length :' + b.length);

  });

}).listen(8000, '127.0.0.1');
```

Node

```
var http    = require('http'),
    request = require('request');

http.createServer(function (req, res) {
  try {
    request.get('https://nodesource.com', function(e, r, b) {
      throw new Error('Error');
      res.writeHead(200, {'Content-Type': 'text/plain'});
      res.end('Length :' + b.length);

    });
  }
  catch(e) {}

}).listen(8000, '127.0.0.1');
```

Servlet Async

```
@RequestMapping("/lengthAsync")
public @ResponseBody DeferredResult<String> length() throws Exception {

   AsyncHttpClient client = new AsyncHttpClient();
   final DeferredResult<String> result = new DeferredResult();

   try {
      client
        .prepareGet(“http://nodesource.com")
        .execute(new AsyncCompletionHandlerBase() {

            @Override
            public Response onCompleted(Response response) throws Exception {
              if (10>0) throw new Exception(“ERROR");
              int len = response.getResponseBody().length();
              result.setResult(Integer.toString(len));
              return super.onCompleted(response);
            }
        }
      );
   }
   catch(Exception e) {}

   return result;
}
```

NO

```
function getLength(url, cb) {

  request.get(url, function(err, resp, body)) {
    if (err) {
      throw err;
    }

    cb(null, body.length);
  };
}
```

Yes

```
function getLength(url, cb) {

  request.get(url, function(err, resp, body)) {
    if (err) {
      return cb(err);
    }

    cb(null, body.length);
  };
}
```

```
function getLength(url, cb) {

  request.get(url, function(err, resp, body)) {
    if (err) {
      return cb(new Error(err)));
    }

    try {
      var obj = JSON.parse(body);
    }
    catch(e) {
      return cb(new Error('Problem parsing resp'));
    }

    cb(null, obj.name);
  };
}
```

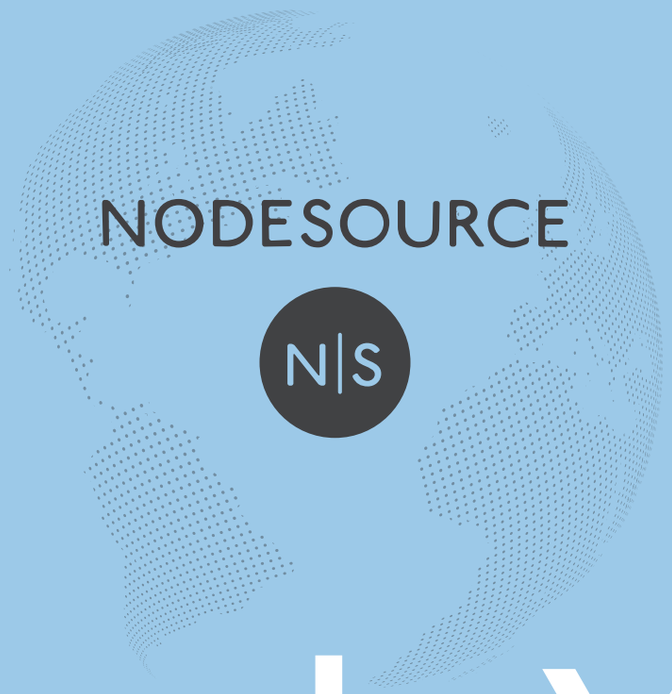
Key Takeaways

- Node solves client concerns, stacks downstream are free focus on data and business logic

- Familiar building blocks, but with some caveats

- Node's community fully embraces asynchronous, event loop based programming

- Limit throwing to truly **exceptional** events



Thank You

ryan@nodesource.com
@ryan_stevens

Special thanks to
Tobias Wennergren, LOYAL3
Sam Douglas, Pandora