JavaOne™

ORACLE®

HTTP 2 Comes to Java

What Servlet 4.0 Means to You

Ed Burns and Shing Wai Chan
Java EE Specification Team
Oracle

JavaOne ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Our Plan for Your Time Investment

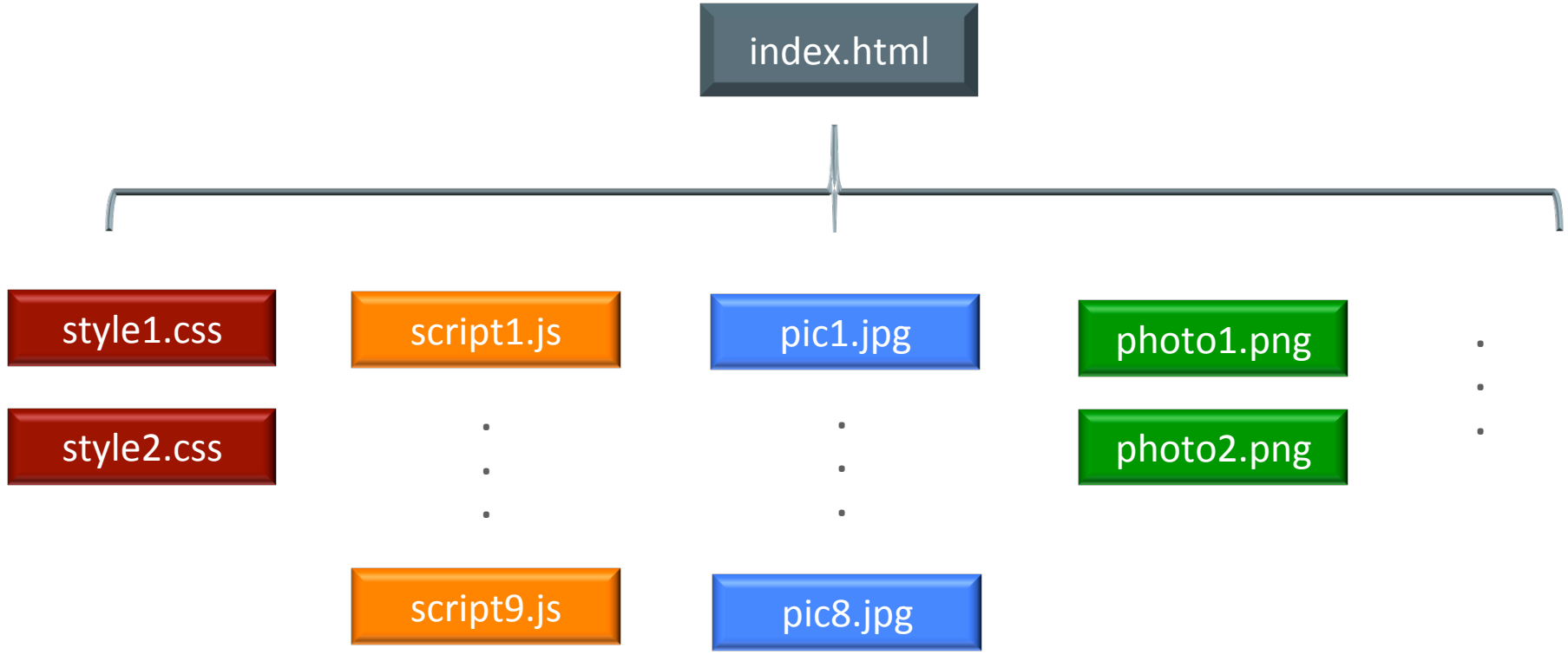
1. Why HTTP 2?
2. HTTP 2 Big Features
3. How Servlet Might Expose These Features
4. Java SE 9 Support for HTTP 2
5. Summary and Current Status

Our Plan for Your Time Investment

1. Why HTTP 2?
2. HTTP 2 Big Features
3. How Servlet Might Expose These Features
4. Java SE 9 Support for HTTP 2
5. Summary and Current Status

Why HTTP 2?

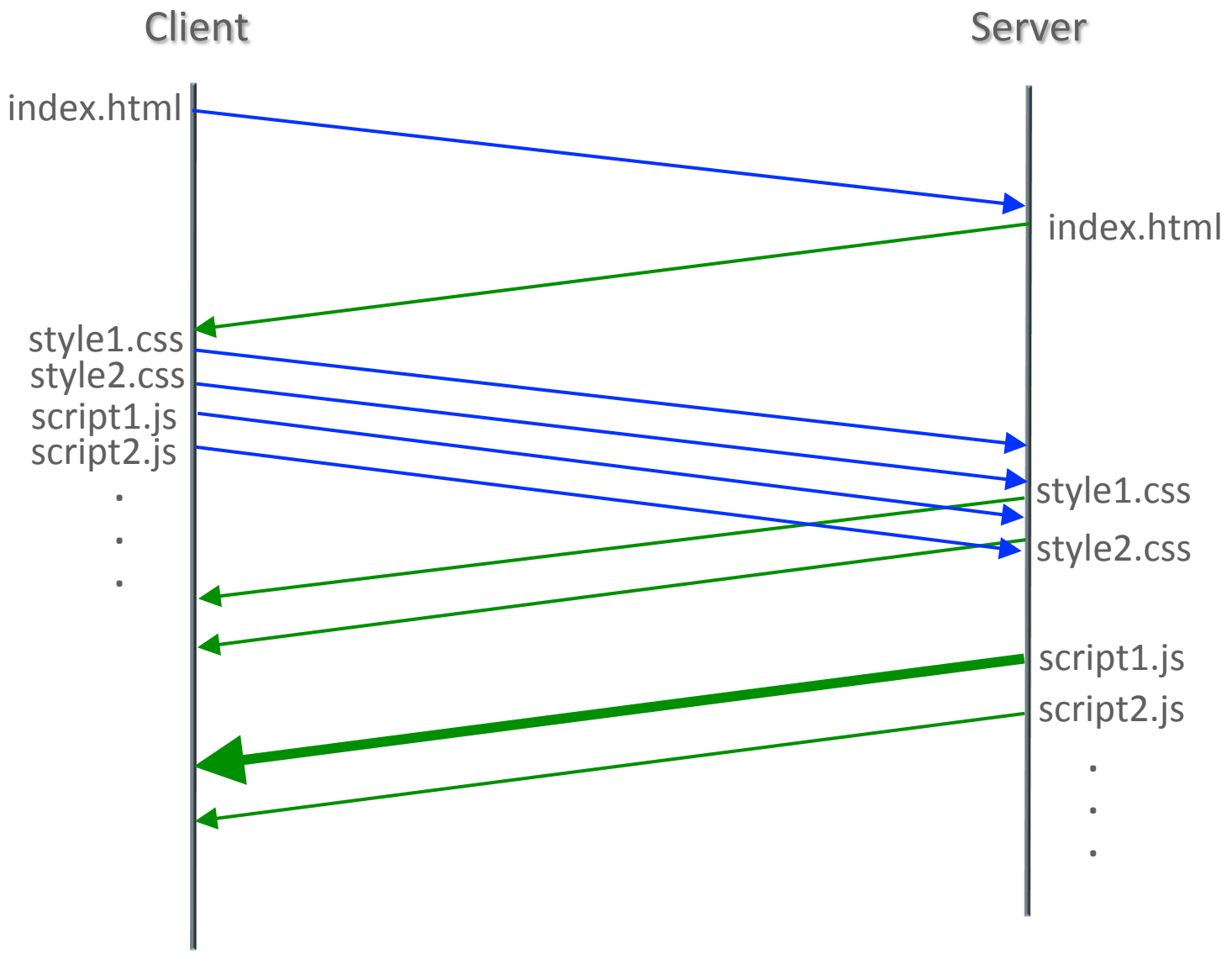
A Real Life Example



Why HTTP 2?

Problems in HTTP 1.1

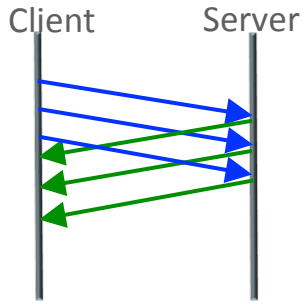
- HTTP Pipelining
- Head-of-Line blocking

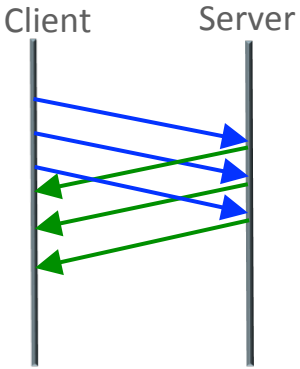


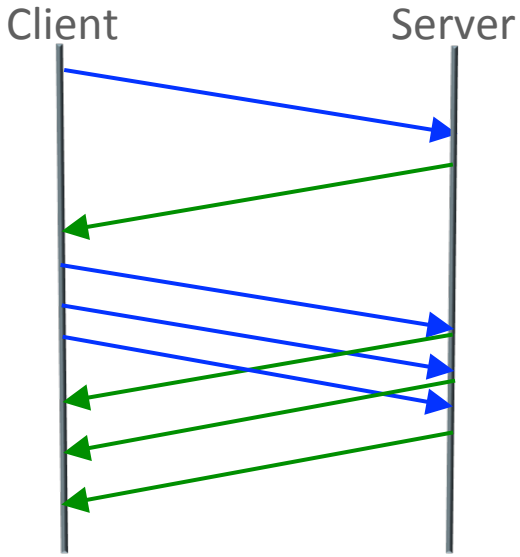
Why HTTP 2?

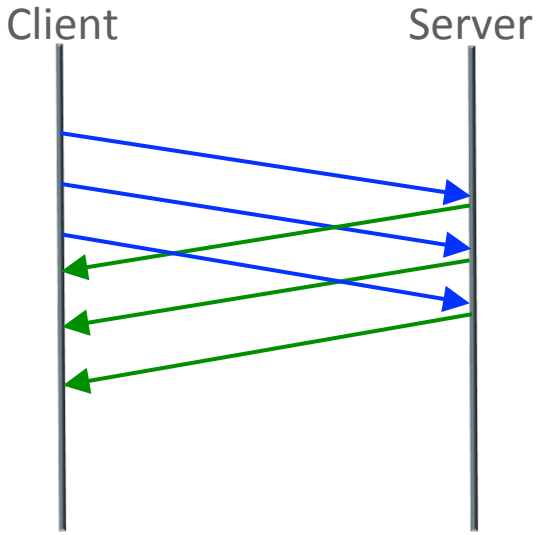
Problems in HTTP 1.1

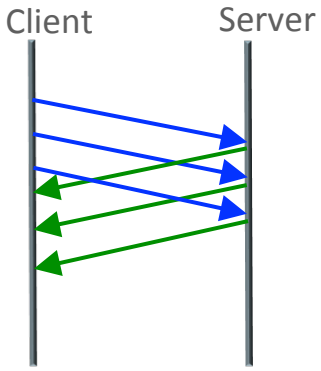
- Inefficient use of TCP sockets

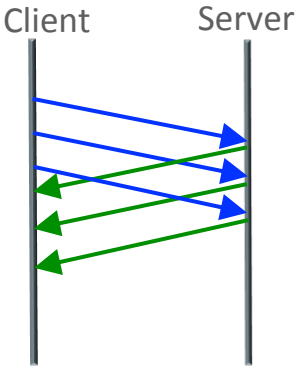












Why HTTP 2?

What is an optimization?

- Much of what we do in web-apps is a hack to work around shortcomings in HTTP 1.1
  - File concatenation and image sprites
  - Domain sharding
  - Inlined assets

File Concatenation and Image Sprites

TCP Efficiency Improves with Larger Files

File Concatenation and Image Sprites

TCP Efficiency Improves with Larger Files

- Modern web page now consists of more than 90 resources fetched from 15 distinct hosts
- Solution:
  - Just work around it by shoving more than one logical file into one physical file.
  - Seminal article: A List Apart **http://alistapart.com/article/sprites**
  - Useful tool: SpritePad **http://spritepad.wearekiss.com/**

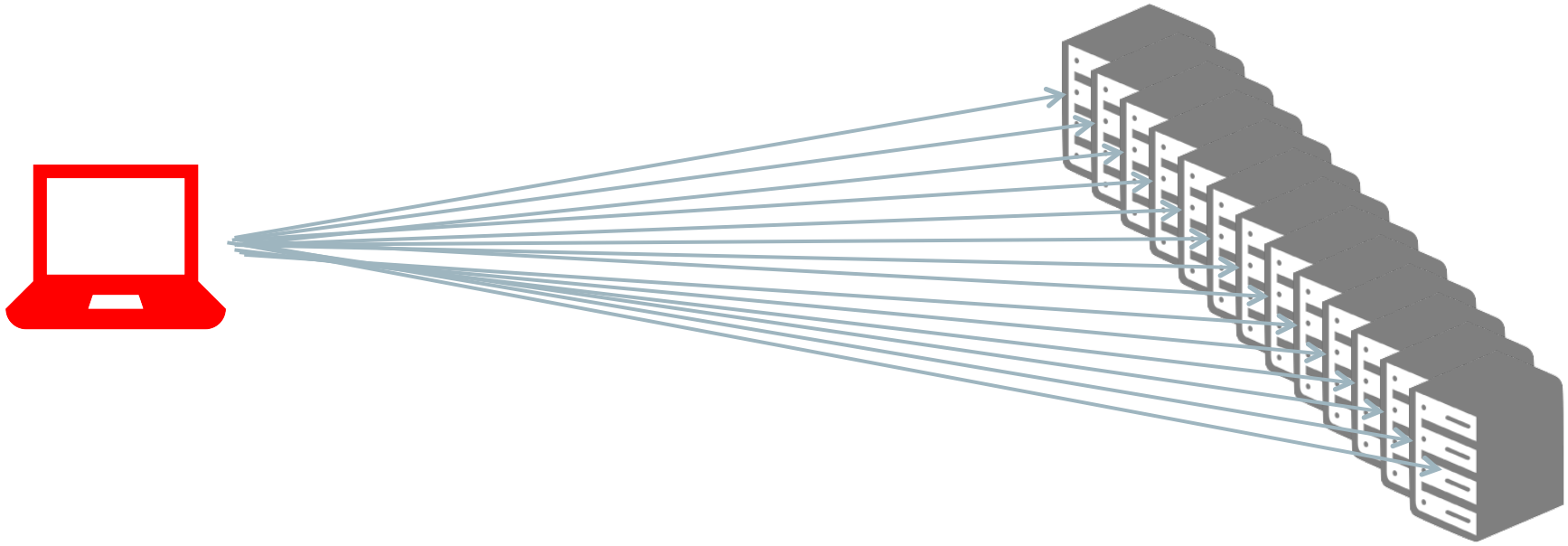
File Concatenation and Image Sprites

TCP Efficiency Improves with Larger Files

```
.ic-AerospaceAndDefense-wht-on-gray, .ic-AerospaceAndDefense-wht-on-red, .ic-Airline-wht-on-gray, .ic-Airline-wht-on-red{
        background: url(sprites.png) no-repeat;
}
.ic-AerospaceAndDefense-wht-on-gray{
        background-position: 0 0;
        width: 80px;
        height: 80px;
}
.ic-AerospaceAndDefense-wht-on-red{
        background-position: -81px 0;
        width: 80px;
        height: 80px;
}
.ic-Airline-wht-on-gray{
        background-position: 0 -80px ;
        width: 80px;
        height: 80px;
}
.ic-Airline-wht-on-red{
        background-position: -81px -79px ;
        width: 80px;
        height: 80px;
}
```

Domain Sharding

Split page resources across several hosts to work around browser limits



Inlined Assets

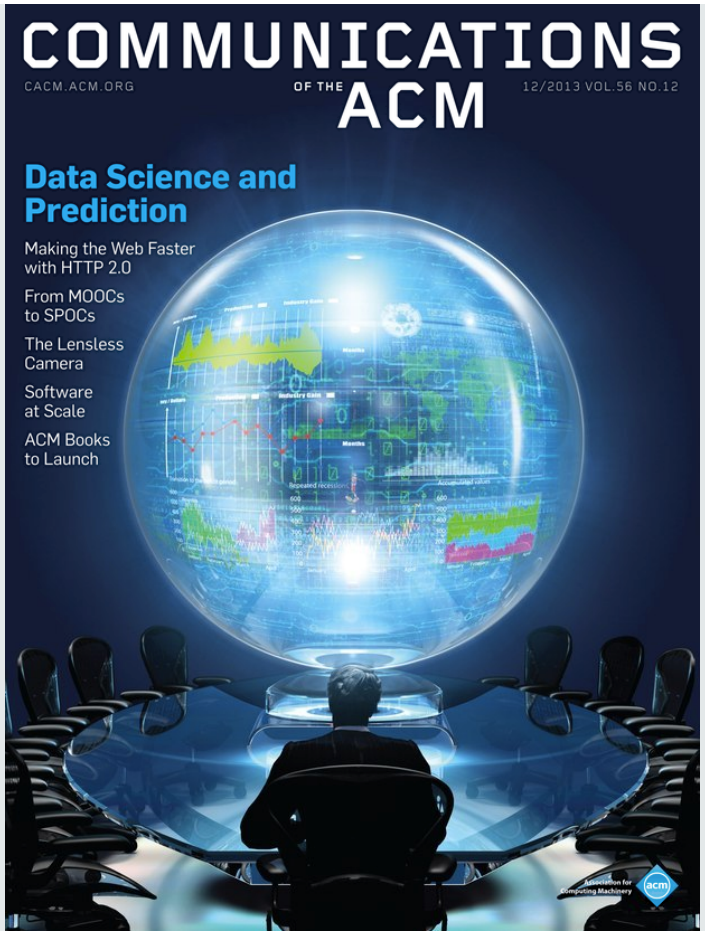
Base64 Encoding Will Never Die

- data URLs
-

Our Plan for Your Time Investment

1. Why HTTP 2?
2. HTTP 2 Big Features
3. How Servlet Might Expose These Features
4. Java SE 9 Support for HTTP 2
5. Summary and Current Status

HTTP/2 Big Ticket Feature Review



- Request/Response multiplexing
- Binary Framing
- Stream Prioritization
- Server Push
- Header Compression
- Upgrade from HTTP 1.1
  - ALPN (or NPN)
  - 101 Switching Protocols

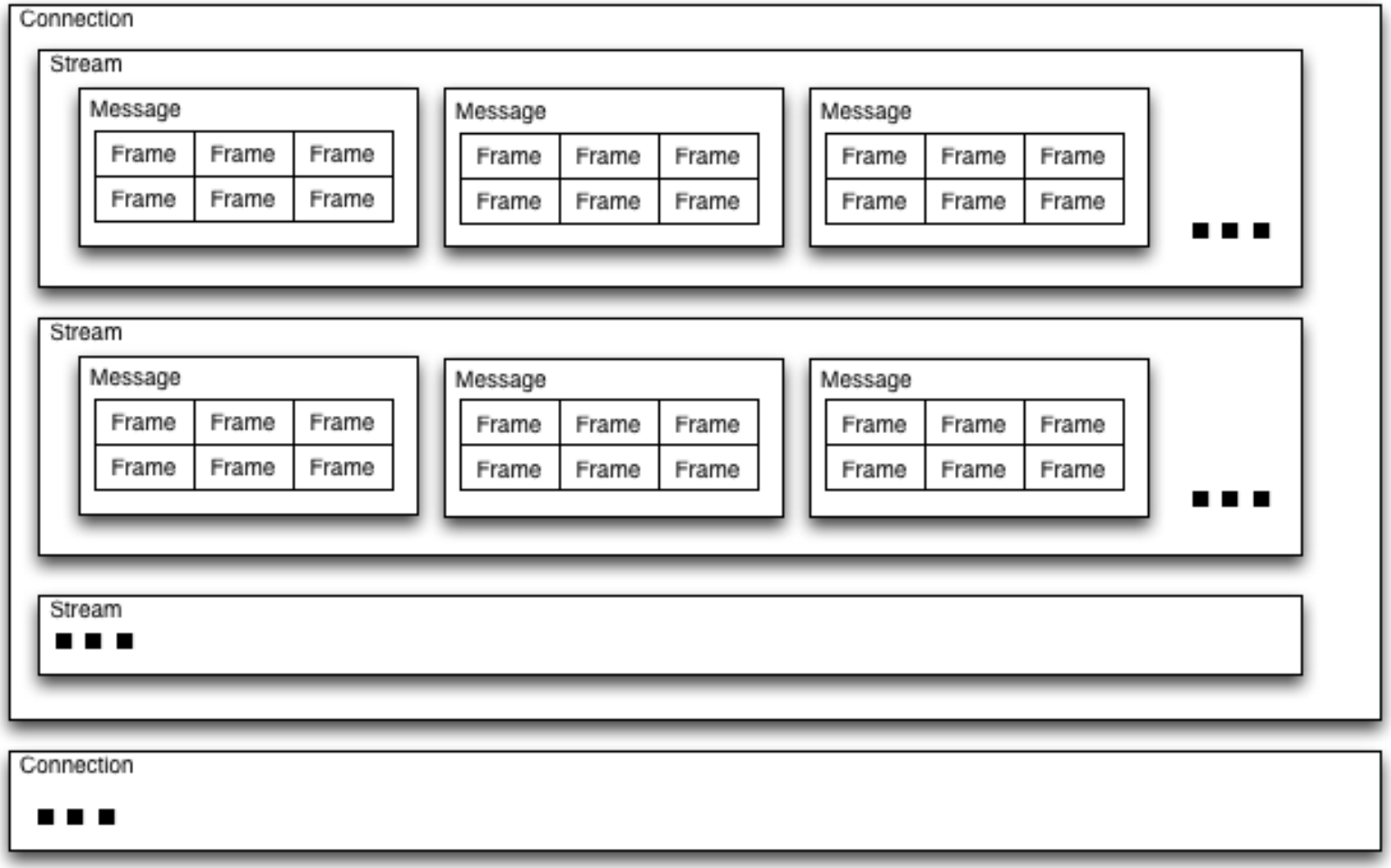
HTTP/2 Request Response Multiplexing

Lets you do more things with a single TCP connection

- Fully bi-directional
- Enabled by defining some terms
  - *Connection*
    A TCP socket
  - *Stream*
    A “channel” within a connection
  - *Message*
    A logical message, such as a request or a response
  - *Frame*
    The smallest unit of communication in HTTP/2.

HTTP/2 Request Response Multiplexing

Connections, Streams, Messages, Frames



HTTP/2 Request Response Multiplexing

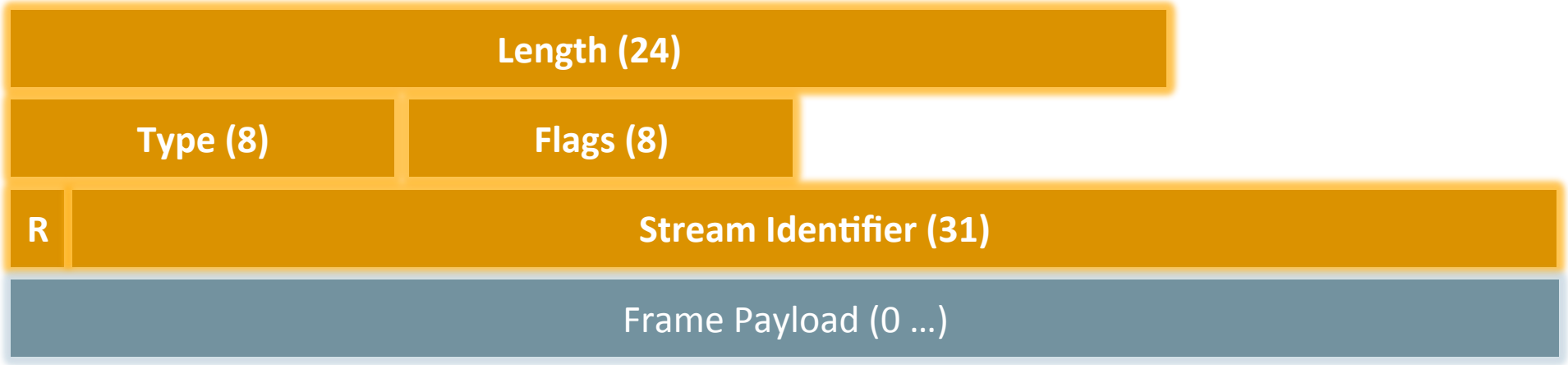
Connections, Streams, Messages, Frames

- Once you break the communication down into frames, you can interweave the logical streams over a single TCP connection.
- Yet another idea from the 1960s is new again.

HTTP/2 Binary Framing

Enabled by dumping newline delimited ASCII

- Solves Head-Of-Line (HOL) blocking problem



- Type field can be DATA, HEADERS, PRIORITY, RST_STREAM, SETTINGS, PUSH_PROMISE, PING, GOAWAY, WINDOW_UPDATE, CONTINUATION

HTTP/2 Binary Framing

Example 1

```
GET /index.html HTTP/1.1
Host: example.com
Accept: text/html
```

```
HEADERS
   + END_STREAM
   + END_HEADERS
      :method: GET
      :scheme: http
      :path: /index.html
      :authority: example.org
      accept: text/html
```

HTTP/2 Binary Framing

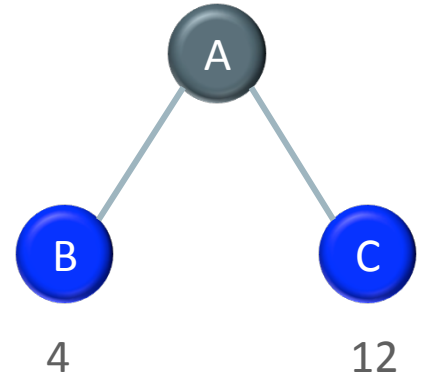
Example 2

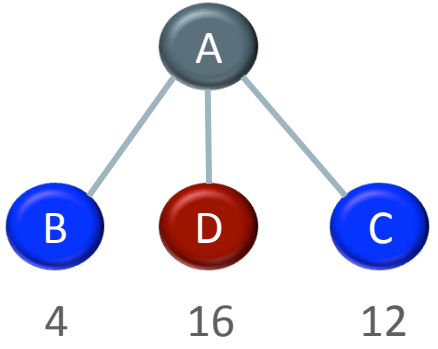
```
HTTP/1.1 200 OK
Content-Length: 11
Content-Type: text/html

Hello World
```

```
HEADERS
   - END_STREAM
   + END_HEADERS
      :status: 200
      content-length: 11
      content-type: text/html

DATA
   + END_STREAM
Hello World
```

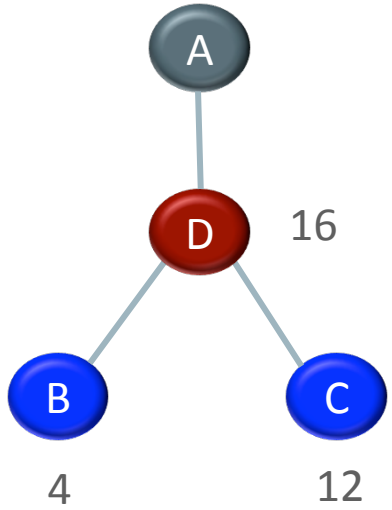
HTTP/2 Stream Prioritization

- Stream Dependency in HEADERS Frame
- PRIORITY frame type
- An additional 40 bytes
  - Stream id (31)
  - Weight (8): [1, 256]
  - Exclusive bit (1)
- Only an advice





exclusive = 0



exclusive = 1

S

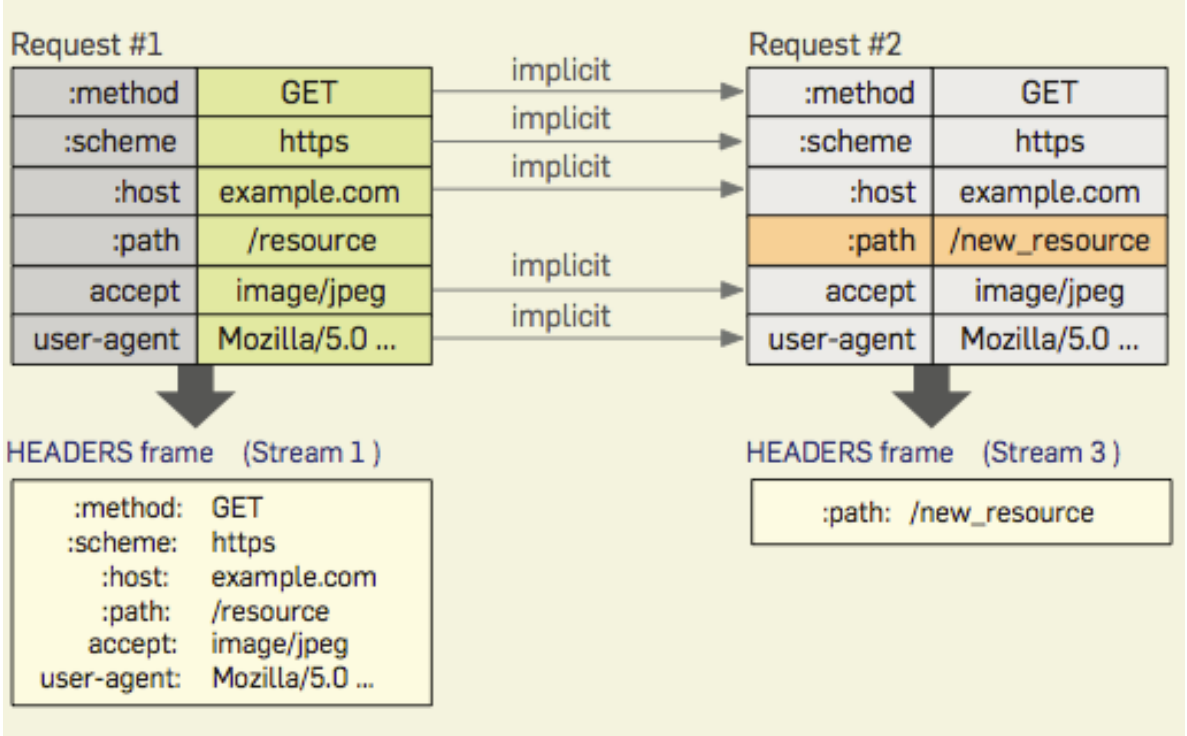
HTTP/2 Server Push

- Eliminates the need for resource inlining.
- Lets the server populate the browser's cache in advance of the browser asking for the resource to put in the cache.
- No corresponding JavaScript API, but can be combined with SSE
  - Server pushes stuff into the browser's cache.
  - Server uses SSE to tell the browser to go fetch it (but we know it's already in the browser's cache).

HTTP/2 Header Compression

Known as HPACK

- Observation: most of the headers are the same in a given stream
  - Host: Accept: user-agent: etc.
- Why send them every time?
- Have the server and the client keep tables of headers, then just send references and updates to the tables.



HTTP/2 Upgrade from HTTP 1.1

Secure or not-secure?

- Not secure
  - We have to use port 80
  - Use existing 101 Switching Protocols from HTTP 1.1
- Secure
  - Next Protocol Negotiation (NPN)
  - Application Layer Protocol Negotiation

Our Plan for Your Time Investment

1. Why HTTP 2?
2. HTTP 2 Big Features
3. How Servlet Might Expose These Features
4. Java SE 9 Support for HTTP 2
5. Summary and Current Status

Servlet 4.0 Big Ticket New Features

Challenges in Exposing HTTP/2 Features in Servlet API

- Existing API is designed for One Request == One Response.
- HTTP/2 destroys this assumption.
- It will be challenging to do justice to the new reality of One Request == One or More Responses.
- We must not simply bolt the “One or More Responses” concept onto some convenient part of the existing API.

Servlet 4.0 Big Ticket New Features

HTTP/2 Features Potentially Exposed in Servlet API

- Request/Response multiplexing
- Binary Framing
- Stream Prioritization
- Server Push
- Header Compression
- Upgrade from HTTP 1.1
  - ALPN or (NPN)
  - 101 Switching Protocols

Servlet 4.0 Big Ticket New Features

HTTP/2 Features Potentially Exposed in Servlet API

- Add method `HttpServletRequest` and `HttpServletResponse`
  - `int getStreamId()`

Servlet 4.0 Big Ticket New Features

Stream Prioritization

- Add a new class `Priority`
  - `boolean exclusive`
  - `int streamId`
  - `int weight`
- Add method to `HttpServletRequest`
  - `Priority getPriority()`
- Add methods to `HttpServletResponse`
  - `Priority getPriority()`
  - `void setPriority(Priority p)`

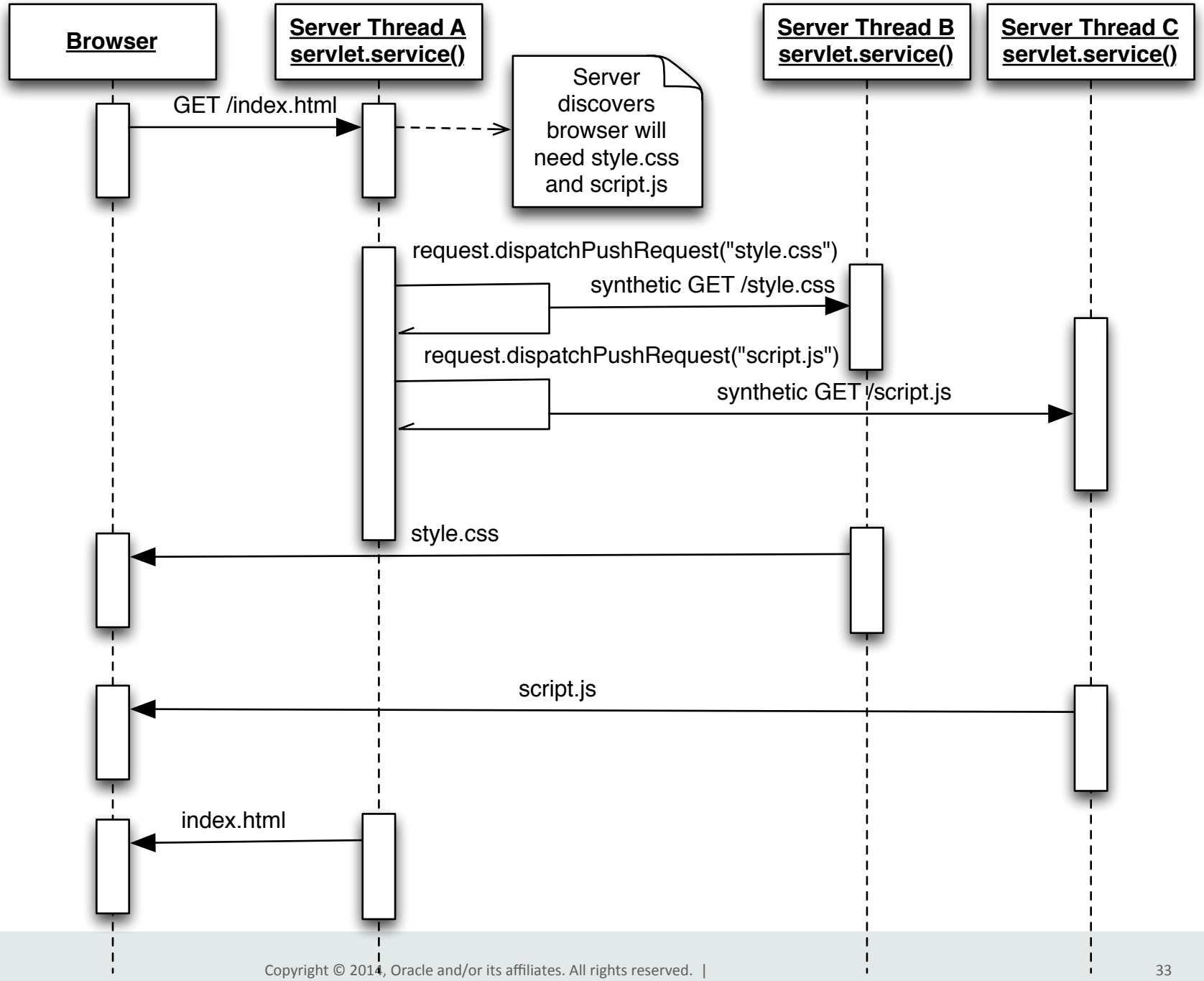
Servlet 4.0 Big Ticket New Features

Server Push

- Push resource to client for a given url and headers
- May add callback for completion or error of a push
- Not at all a replacement for WebSocket
- Really useful for frameworks that build on Servlet, such as JSF

Servlet 4.0 Big Ticket New Features

Server Push



Server Push

Example of Potential Use from JSF

```
public class FacesServlet implements Servlet {
public void service(ServletRequest req,
                    ServletResponse resp) throws IOException, ServletException {
    //..
    HttpServletRequest request = (HttpServletRequest) req;
    try {
        ResourceHandler handler =
            context.getApplication().getResourceHandler();
        if (handler.isResourceRequest(context) || request.isPushRequest()) {
            handler.handleResourceRequest(context);
        } else {
            lifecycle.attachWindow(context);
            lifecycle.execute(context);
            lifecycle.render(context);
        }
    }
}
```

Server Push

Example of Potential Use from JSF

```
public class ExternalContextImpl extends ExternalContext {
    //…
    public String encodeResourceURL(String url) {
        if (null == url) {
            String message = MessageUtils.getExceptionMessageString
                (MessageUtils.NULL_PARAMETERS_ERROR_MESSAGE_ID, "url");
            throw new NullPointerException(message);
        }
        Map attrs = getResourceAttrs();
        ((HttpServletRequest) request).dispatchPushRequest(url, attrs);
        return ((HttpServletResponse) response).encodeURL(url);
    }
    //…

}
```

Reactive Programming

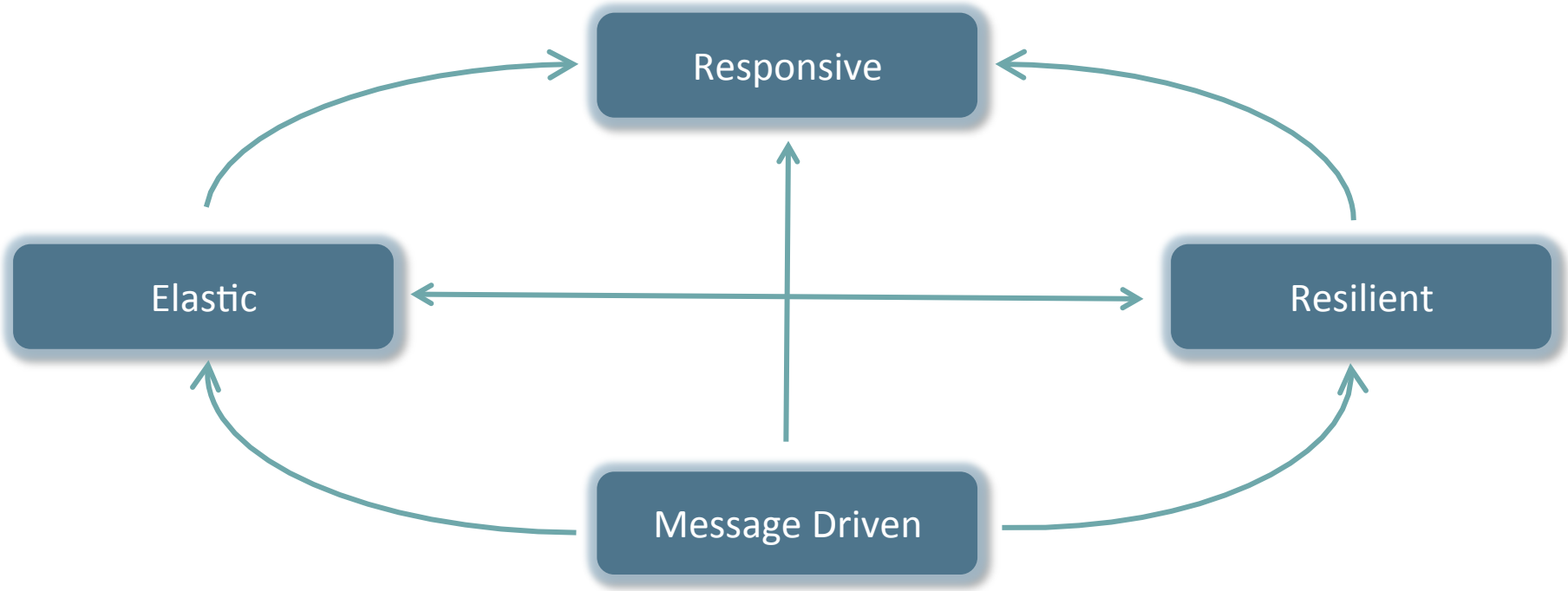


image credit: reactivemanifesto.org

S

Servlet 4.0 and Reactive Programming

- Non-blocking IO in Servlet 3.1
  - `ServletInputStream`
    - `#setReadListener, #isReady`
  - `ServletOutputStream`
    - `#setWriteListener, #isReady`
  - `ReadListener`
    - `#onDataAvailable, #onAllDataRead, #onError`
  - `WriteListener`
    - `#onWritePossible, #onError`

Servlet 4.0 and Reactive Programming

- Asynchronous in Servlet 3.0
  - `ServletRequest#startAsync`
  - `AsyncContext`
    - `#addListener, #dispatch, #complete`
  - `AsyncListener`
    - `#onComplete, #onError, #onStartAsync, #onTimeout`
- Event-driven
  - Server-Sent Events

Our Plan for Your Time Investment

1. Why HTTP 2?
2. HTTP 2 Big Features
3. How Servlet Might Expose These Features
4. Java SE 9 Support for HTTP 2
5. Summary and Current Status

Java SE 9 Support for HTTP/2

- JEP 110 http://openjdk.java.net/jeps/110
- Easy to use API
- Covers only the most common use cases
- Supports both HTTP 1.1 and 2

Java SE 9 Support for HTTP/2

Small footprint

- Two classes (working titles)
  - HttpRequestGroup
    - configuration for multiple requests
  - HttpRequest
    - one request/response interaction

Java SE 9 Support for HTTP/2

Small footprint

- Blocking mode: one thread per request/response
  - send request
  - get response
- Non-blocking mode: handle multiple request/response interactions in single thread using non-blocking API
  - analogous to NIO selectors

Java SE 9 Support for HTTP/2

Callback handlers

- Responses handled through lambda style callback handlers
  - invoked on the calling thread
  - Must allocate threads to responses manually
- Request/Response bodies handled as InputStream/OutputStream

Java SE 9 Support for HTTP/2

```
HttpRequestGroup group = HttpRequestGroup.create();
HttpRequest req = group.createRequest()
   .setRequestMethod("POST")
   .setRequestURI(new URI("http://www.foo.com/a/b")
   .setRequestBody("Param1=1,Param2=2")
   .onResponseHeader("X-Foo", (request, name, value) -> {
       System.out.printf(" received an X-Foo header");
     })
   .sendRequest()
   .waitForCompletion();
```

Java SE 9 Support for HTTP/2

```
HttpRequestGroup group = HttpRequestGroup.create();
HttpRequest req = group.createRequest() …
  .onResponseBody((HttpRequest request, InputStream in) -> {
    if (request.getResponseCode() == 200) {
      Path out = Paths.get("/tmp/out");
      try { Files.copy(in, out); } finally { in.close(); }
    }
   })
  .sendRequest()
  .waitForCompletion();
```

Java SE 9 Support for HTTP/2

HTTP/2 features

- Negotiation of HTTP/2 from 1.1
  - ALPN or plaintext
- Server Push
  - Support for PUSH_PROMISE frames
- HPACK parameters

Our Plan for Your Time Investment

1. Why HTTP 2?
2. HTTP 2 Big Features
3. How Servlet Might Expose These Features
4. Java SE 9 Support for HTTP 2
5. Summary and Current Status

Summary and Current Status

- Servlet 4.0 brings HTTP 2 to Java EE
  - 100% compliant implementation of HTTP 2
  - Expose key features to the API
    - Server Push
    - Stream Prioritization
    - Request/Response multiplexing

Summary and Current Status

- JSR-369 just formed on 22 September
- Tentative Delivery Schedule
  - Q3 2014: expert group formed
  - Q1 2015: early draft
  - Q3 2015: public review
  - Q4 2015: proposed final draft
  - Q3 2016: final release

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

JavaOne™

ORACLE®