

NODE PROGRAM

EXPRESS.JS



NODE.JS VERSION: 5.1
LAST UPDATED: JAN 2016

EXPRESS

**EXPRESS IS THE MOST POPULAR WEB APPLICATION FRAMEWORK
FOR NODE**

**IT IS EASY TO WORK WITH AS IT TIES INTO NODE'S FUNCTIONAL
PARADIGM**

- > DELIVER STATIC CONTENT (OR CONSIDER USING NGINX)**
 - > MODULARIZE BUSINESS LOGIC**

INSTALLING DEPENDENCY

```
$ npm install express --save
```

INSTALLING SCAFFOLDING

INSTALL EXPRESS.JS COMMAND-LINE GENERATOR:

```
$ npm install -g express-generator
```

USING THE GENERATOR

```
$ express todo-list-app
```

- **app.js: MAIN FILE, HOUSES THE EMBEDDED SERVER AND APPLICATION LOGIC**
- **public/: CONTAINS STATIC FILES TO BE SERVED BY THE EMBEDDED SERVER**
- **routes/: HOUSES CUSTOM ROUTING FOR THE EMBEDDED**

CONFIGURING EXPRESS

THE EXPRESS SERVER NEEDS TO BE CONFIGURED BEFORE IT CAN START

MANAGE CONFIGURATION VIA THE `set` METHOD:

```
var app = express();  
app.set('port', process.env.PORT || 3000);  
app.set('views', 'views'); // the directory the templates are stored in  
app.set('view engine', 'jade');
```

NODE.JS MIDDLEWARE PATTERN

WHAT IS MIDDLEWARE

MIDDLEWARE PATTERN IS A SERIES OF PROCESSING UNITS CONNECTED TOGETHER, WHERE THE OUTPUT OF ONE UNIT IS THE INPUT FOR THE NEXT ONE. IN NODE.JS, THIS OFTEN MEANS A SERIES OF FUNCTIONS IN THE FORM:

```
function(args, next) {  
    next(output) // error or real output  
}
```


CONNECT MIDDLEWARE

EXAMPLE:

```
app.use(function middleware1(req, res, next) {  
  // middleware 1  
  next();  
});  
app.use(function middleware2(req, res, next) {  
  // middleware 2  
  next();  
});
```

MIDDLEWARE ORDER

MIDDLEWARE ARE EXECUTED IN THE ORDER SPECIFIED:

```
var logger = require('morgan')
var bodyParser = require('body-parser')
...
app.use(logger('dev'))
app.use(bodyParser.json())
```

CREATING MIDDLEWARE

CUSTOM MIDDLEWARE IS EASY TO CREATE:

```
app.use(function (req, res, next) {  
  // modify req or res  
  // execute the callback when done  
  next();  
});
```

CONNECT FRAMEWORK

**EXPRESS LEVERAGES THE CONNECT FRAMEWORK TO PROVIDE
MIDDLEWARE
FUNCTIONALITY.**

**MIDDLEWARES ARE USED TO MANAGE HOW A REQUEST SHOULD BE
HANDLED.**

MOST POPULAR AND USEFUL CONNECT/EXPRESS MIDDLEWARE

```
$ npm install <package_name> --save
```

- BODY-PARSER REQUEST PAYLOAD
 - COMPRESSION GZIP
- CONNECT-TIMEOUT SET REQUEST TIMEOUT
 - COOKIE-PARSER COOKIES
- COOKIE-SESSION SESSION VIA COOKIES STORE

CONNECT/EXPRESS MIDDLEWARE

- CSURF CSRF
- ERRORHANDLER ERROR HANDLER
- EXPRESS-SESSION SESSION VIA IN-MEMORY OR OTHER STORE
 - METHOD-OVERRIDE HTTP METHOD OVERRIDE
 - MORGAN SERVER LOGS
 - RESPONSE-TIME

CONNECT/EXPRESS MIDDLEWARE

- SERVE-FAVICON FAVICON
 - SERVE-INDEX
- SERVE-STATIC STATIC CONTENT
 - VHOST

OTHER POPULAR MIDDLEWARE

- COOKIES AND KEYGRIP: ANALOGOUS TO `cookieParser`
 - RAW-BODY
 - CONNECT-MULTIPARTY, CONNECT-BUSBOY
 - QS: ANALOGOUS TO `query`
- ST, CONNECT-STATIC ANALOGOUS TO `staticCache`

OTHER POPULAR MIDDLEWARE

- EXPRESS-VALIDATOR: VALIDATION
 - LESS: LESS CSS
- PASSPORT: AUTHENTICATION LIBRARY
 - HELMET: SECURITY HEADERS
 - CONNECT-CORS: CORS
 - CONNECT-REDIS

TEMPLATE ENGINE

SETTING THE `view engine` VARIABLE TO `jade` FOR INSTANCE.
WOULD TRIGGER
THE FOLLOWING FUNCTION CALL INTERNALLY

```
app.set('view engine', 'jade'); // shorthand  
  
// does the same as the above  
app.engine('jade', require('jade').__express);
```

TEMPLATE ENGINE

CUSTOM CALLBACKS CAN BE DEFINED TO PARSE TEMPLATES

```
app.engine([format], function (path, options, callback) {  
  // template parsing logic goes here  
});
```

**NOTE: CUSTOM CALLBACKS ARE USEFUL IF THE TEMPLATE ENGINE
DOESN'T EXPORT
AN __EXPRESS FUNCTION**

RUNNING EXPRESS

```
var http = require('http'),  
    express = require('express');  
  
var app = express();  
  
// ...  
  
var server = http.createServer(app);  
server.listen(app.get('port'), function () {  
    // Do something... maybe log some info?  
});
```

DEMO




RESTFUL API

CORE HTTP MODULE API: [HTTP://BIT.LY/1STXFSG](http://bit.ly/1stxfsg)

EXPRESS: [HTTPS://GITHUB.COM/AZAT-CO/REST-API-EXPRESS](https://github.com/azat-co/rest-api-express)

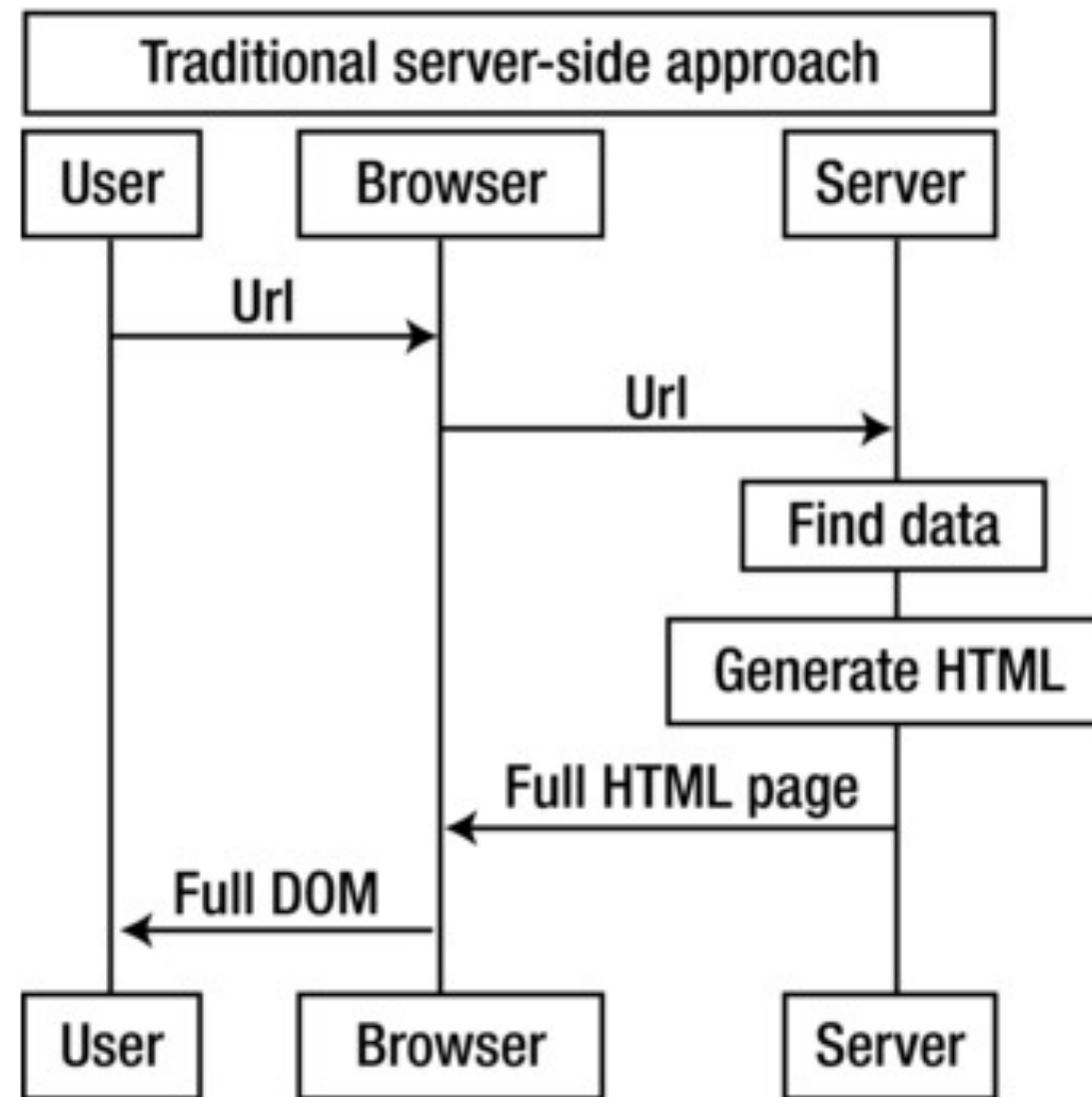
ALTERNATIVES

- > SAILS
- > LOOPBACK 
- > METEOR
- > HAPI
- > RESTIFY

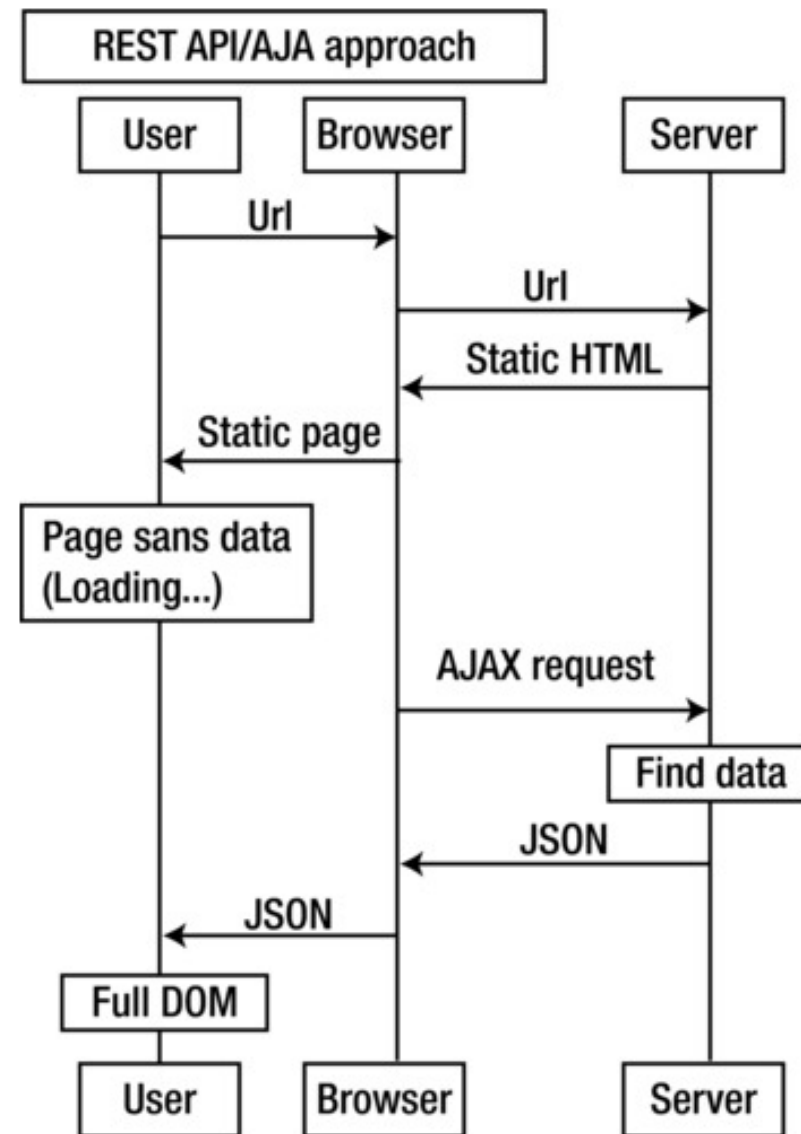
REGISTRY OF HAND-PICKED NODE FRAMEWORKS: NODEFRAMEWORK.COM

BUILDING A RESTFUL API

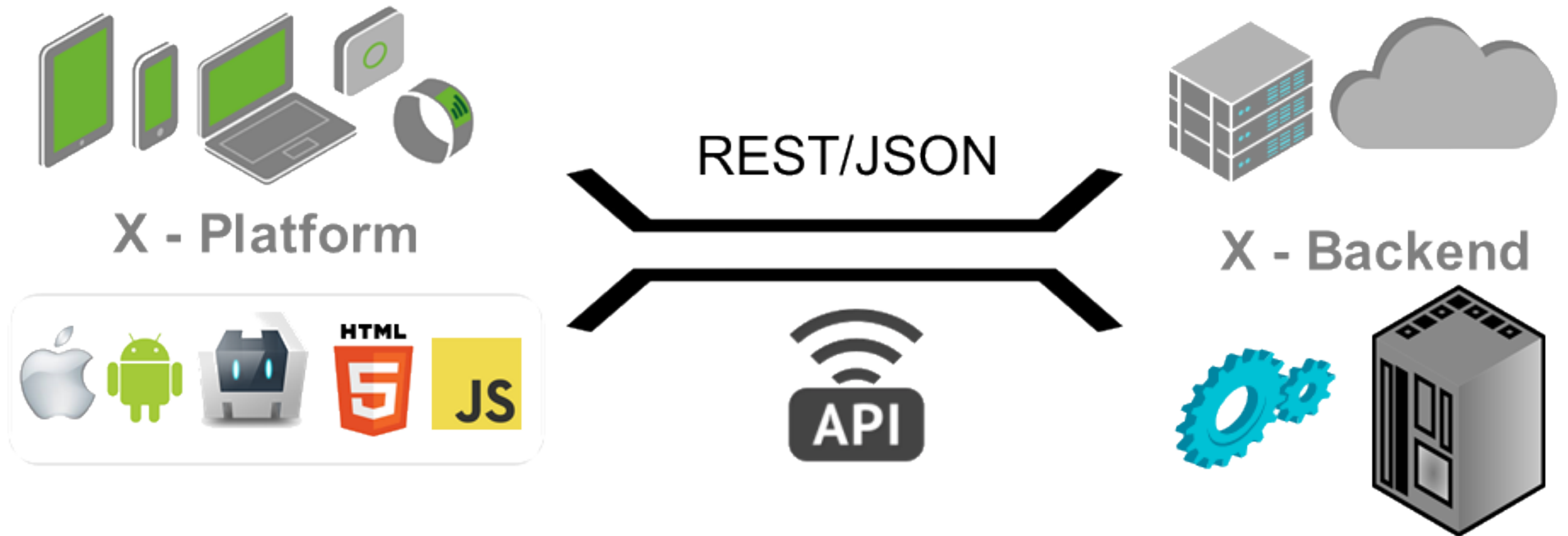
TRADITIONAL WEB APP



API + AJAX/XHR WEB APP

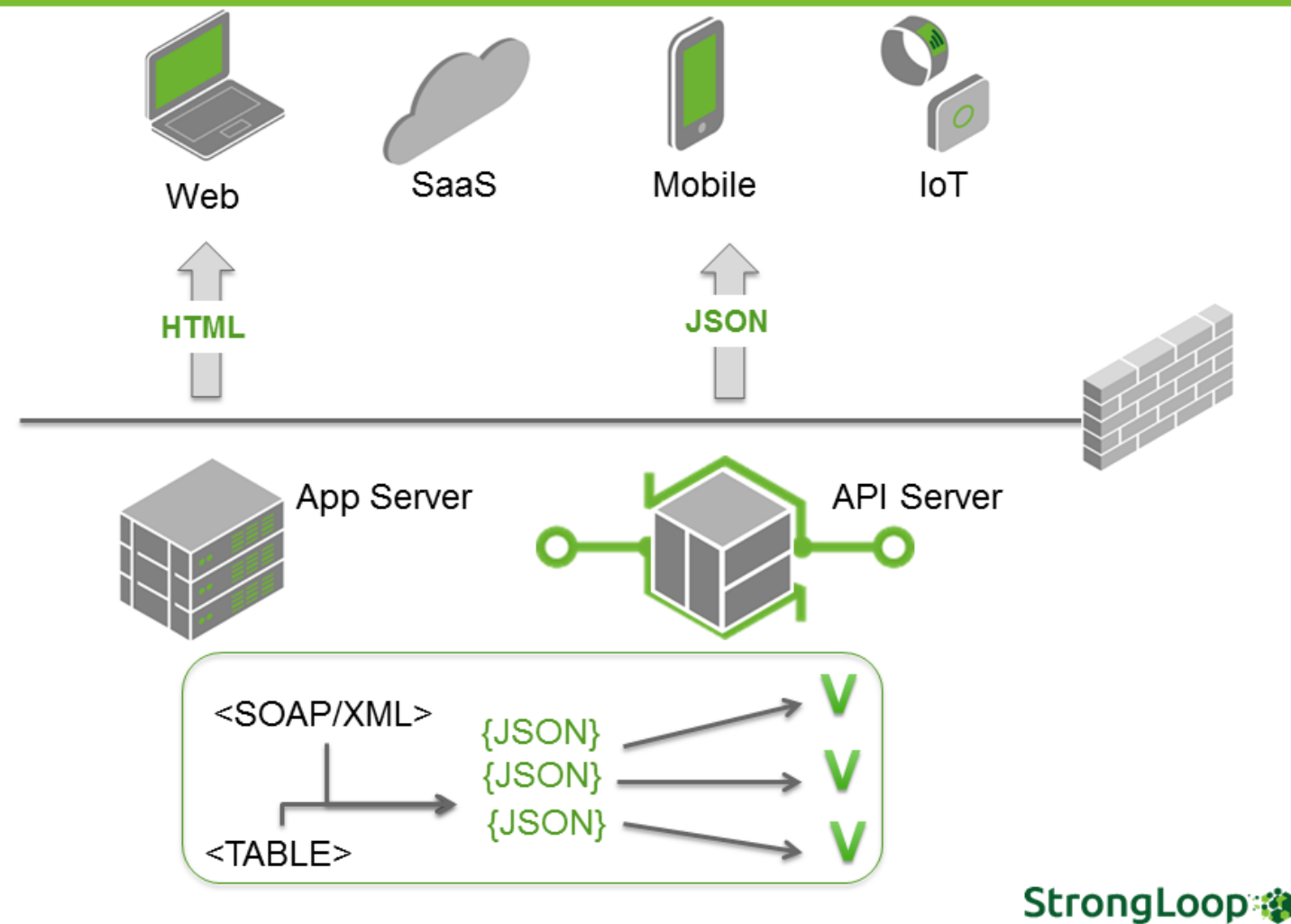


NODE, SPAS AND REST



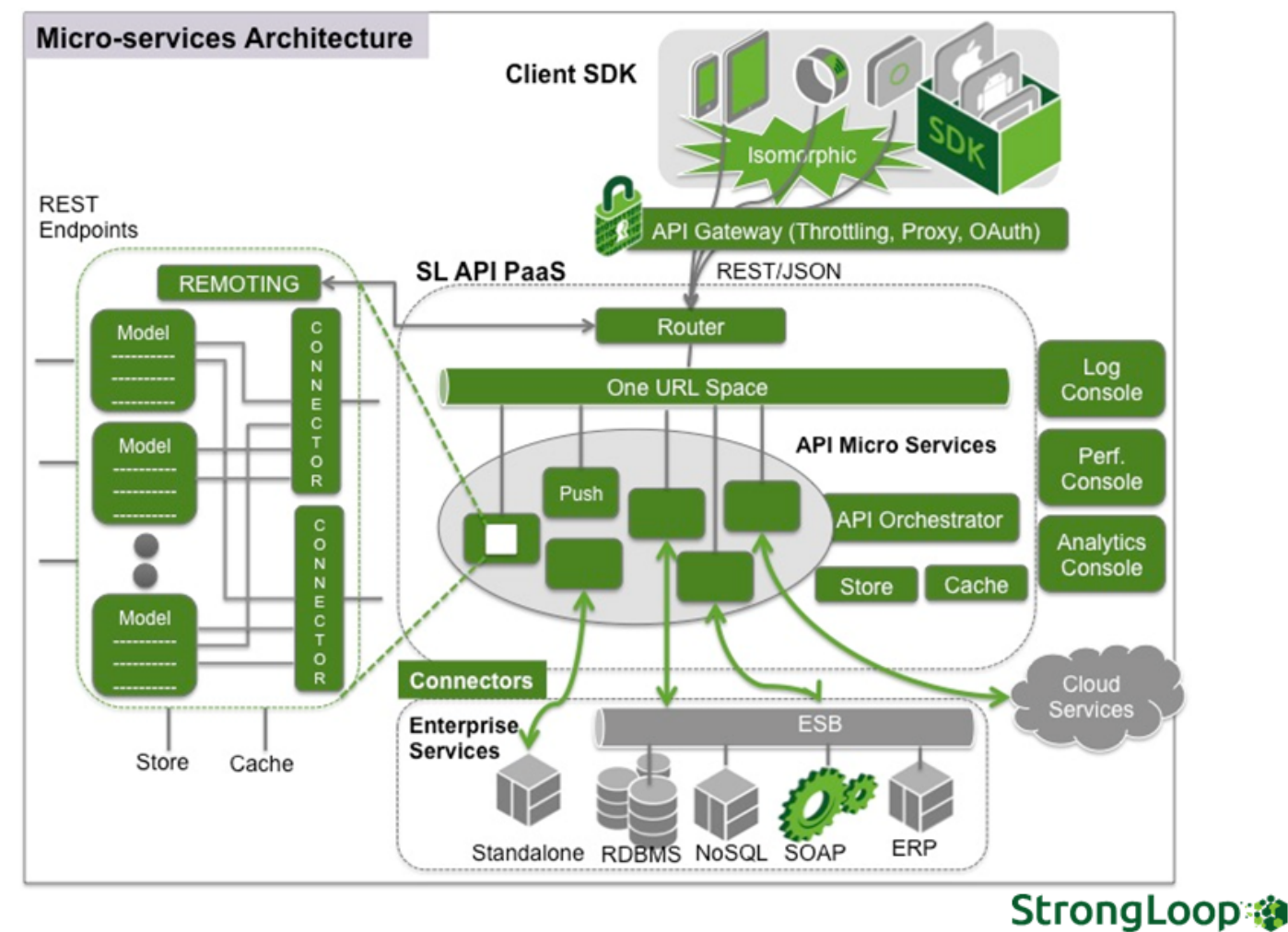
API DECOMPOSITION

API “Decomposition” is the game changer



MICROSERVICES

Micro-services have arrived



REST BASICS

**REPRESENTATIONAL STATE TRANSFER (REST) IS AN
ARCHITECTURAL PATTERN FOR DEVELOPING NETWORK
APPLICATIONS**

**REST SYSTEMS AIM TO KEEP THINGS SIMPLE WHEN CONNECTING
TO AND EXCHANGING DATA BETWEEN MACHINES**

WHY HTTP?

HTTP IS THE IDEAL PROTOCOL FOR REST, GIVEN ITS STATELESS NATURE AND CLIENT-SERVER ARCHITECTURE

- REST IS FAR SIMPLER COMPARED TO REMOTE PROCEDURE CALLS (RPC) AND WEB SERVICES (SOAP, UDDI, ETC)**
- RPCS AND WEB SERVICES RELY ON COMPLEX VOCABULARIES FOR COMMUNICATION**
 - EACH NEW OPERATION IS A NEW VOCABULARY ENTRY.**

REST VERBS

REST USES HTTP REQUESTS (AND VERBS) FOR CRUD OPERATIONS

- **GET**
- **PUT**
- **POST**
- **DELETE**

REST VERBS

AND SOMETIMES...

- PATCH
- HEAD
- OPTIONS

COMMON ENDPOINTS

GET	/tickets	- Retrieve a list of tickets
GET	/tickets/12	- Retrieve a specific ticket
POST	/tickets	- Create a new ticket
PUT	/tickets/12	- Update ticket #12
DELETE	/tickets/12	- Delete ticket #12
PATCH	/tickets/12	- Partially update ticket #12
OPTIONS	/tickets/12	- What can I do to ticket #12?
HEAD	/tickets/12	- What headers would I get if I tried to get ticket #12?

'RESOURCES'

**RESOURCES ARE ENTITIES THAT CAN BE STORED ON A COMPUTER.
SUCH AS:**

- > FILES**
- > DATABASE ENTRIES**
- > PROCESSED OUTPUT FROM FUNCTIONS**

'RESOURCES'

**REST USES HTTP REQUESTS AND RESPONSES TO PROVIDE
REPRESENTATIONS OF RESOURCES**

**FOR EXAMPLE, THE CURRENT VERSION OF A FILE AVAILABLE FOR
DOWNLOAD VIA ITS URL IS A REPRESENTATION OF A FILE
RESOURCE**

**MODIFYING A RESOURCE, SUCH AS CHANGING THE CONTENTS OF A
FILE OR DELETING IT, IS ALSO A RESOURCE STATE THAT CAN BE**

EXPRESS EXAMPLES

GET

TO ALLOW RETRIEVAL BY ID...

```
app.get('/users/:id', function (req, res) {  
  var id = req.params.id;  
  // code to retrieve a single user  
  res.send(user);  
});
```

GET

GET HANDLERS CAN ALSO BE USED TO RETRIEVE A COLLECTION OF RESOURCES

```
app.get('/users', function (req, res) {  
  // code to retrieve multiple users  
  res.send(users);  
});
```

POST

TO CREATE A RESOURCE...

```
app.post('/users', function (req, res) {  
  var username = req.body.username;  
  var email = req.body.email;  
  // ...  
  // code to create a new user  
  res.send(user);  
});
```

OR MAYBE JUST SEND BACK THE ENDPOINT TO GET THE USER...

```
res.send('/api/user/' + user.id);
```


PUT

**TO UPDATE A RESOURCE (OR CREATE IF IT DOESN'T EXIST.
PERHAPS)...**

```
app.put('/users/:id', function (req, res) {  
  var id = req.params.id;  
  // check if the user exists  
  ...  
  if (exists) {  
    // code to modify the user  
  } else {  
    // code to create the user  
  }  
  res.send(user);  
});
```

DELETE

TO DELETE A RESOURCE, CREATE A DELETE HANDLER FOR THE DESIRED URI

```
app.delete('/users/:id', function (req, res) {  
  var id = req.params.id;  
  // code to delete the user  
  res.send(user); // or maybe the URL to create a new user?  
});
```

NOTE: `del` IS DEPRECATED.

HTTP REQUESTS

A CLIENT'S HTTP REQUEST IS ACCESSIBLE FROM WITHIN ROUTING HANDLERS

IT IS THE FIRST ARGUMENT IN THE HANDLER'S CALLBACK

```
app.get('/users/:id', function (req, res) {  
  // req is the request object  
});
```

NOTE: ACCESS TO THE REQUEST OBJECT GRANTS INSIGHT INTO THE CLIENT'S HTTP REQUEST, PROVIDING DATA ON THE REQUEST HEADER, BODY, ET AL.

ACCESSING ROUTE PARAMETERS

A URI SEGMENT CAN BE PARAMETERIZED BY PREFIXING IT WITH A SEMI-COLON

```
app.get( '/users/:id/:another/:segment' , function (req, res) { ... } );
```

HANDLERS SIGNATURES

- `function(request, response, next) {}:`
REQUEST HANDLER SIGNATURE
- `function(error, request, response, next)`
`{}:` **ERROR HANDLER SIGNATURE**

URL PARAMETERS

THESE DYNAMIC PARAMETERS CAN THEN BE ACCESSED VIA THE
REQUEST'S **PARAMS** OBJECT

GET /USERS/:ID

```
req.params.id;
```

URL PARAMETERS

GET /USERS/:ID/:SOME/:FILTER

```
req.params.id;  
req.params.some;  
req.params.filter;
```

QUERY STRINGS

EXPRESS CONVERTS A URL'S QUERY STRING INTO JSON

IT CAN BE ACCESSED VIA THE REQUEST'S `QUERY` OBJECT GET
HTTP://LOCALHOST:3000/?NAME=BRUCE
+WAYNE&AGE=40&OCCUPATION=BATMAN

```
req.query.name;           // "Bruce Wayne"  
req.query.age;            // "40"  
req.query.occupation;     // "Batman"
```


REQUEST BODY

**ENABLE THE `json()` AND `urlencoded()` MIDDLEWARE TO
CONVERT RAW FORM DATA INTO JSON**

```
$ npm install body-parser --save
```

REQUEST BODY

IMPORT MIDDLEWARE:

```
var bodyParser = require('body-parser')
```

PARSE application/json

```
app.use(bodyParser.json());
```

PARSE application/x-www-form-urlencoded

```
app.use(bodyParser.urlencoded({extended: false}))
```

ACCESSING FORM DATA

**FORM DATA IS THEN ACCESSIBLE VIA THE REQUEST'S BODY OBJECT
(ULRENCODED)**

```
// POST name=Bruce+Wayne&age=40&occupation=Your+Average+Businessman
```

```
req.body.name;  
req.body.age;  
req.body.occupation;
```

FILE UPLOADS

FILE UPLOADS FROM WEB FORMS (MULTIPART/FORM-DATA) CAN BE PARSED WITH THESE LIBRARIES:

- **[HTTPS://GITHUB.COM/EXPRESSJS/MULTER](https://github.com/expressjs/multer)**
- **[HTTPS://GITHUB.COM/YAHOO/EXPRESS-BUSBOY](https://github.com/yahoo/express-busboy)**
- **[HTTPS://GITHUB.COM/MSCDEX/CONNECT-BUSBOY](https://github.com/mscdex/connect-busboy)**
- **[HTTPS://GITHUB.COM/ANDREWRK/NODE-MULTIPARTY](https://github.com/andrewrk/node-multipart)**

PARSING JSON

PARSE VARIOUS DIFFERENT CUSTOM JSON TYPES AS JSON

```
app.use(bodyParser.json({ type: 'application/*+json' })))
```

PARSING BUFFER

PARSE SOME CUSTOM THING INTO A BUFFER

```
app.use(bodyParser.raw({ type: 'application/vnd.custom-type' })))
```

PARSING HTML

PARSE AN HTML BODY INTO A STRING

```
app.use(bodyParser.text({ type: 'text/html' } ))
```

HTTP VERBS AND ROUTES

- › `app.get(urlPattern, requestHandler[, requestHandler2, ...])`
- › `app.post(urlPattern, requestHandler[, requestHandler2, ...])`
- › `app.put(urlPattern, requestHandler[, requestHandler2, ...])`
- › `app.delete(urlPattern, requestHandler[,`

REQUEST

- › `request.params`: **PARAMETERS MIDDLEWARE**
- › `request.param`: **EXTRACT ONE PARAMETER**
- › `request.query`: **EXTRACT QUERY STRING PARAMETER**
 - › `request.route`: **RETURN ROUTE STRING**
- › `request.cookies`: **COOKIES. REQUIRES COOKIEPARSER**
- › `request.signedCookies`: **SIGNED COOKIES. REQUIRES**

REQUEST HEADER SHORTCUTS

- `request.get(headerKey):` **VALUE FOR THE HEADER KEY**
 - `request.accepts(type):` **CHECKS IF THE TYPE IS ACCEPTED**
- `request.acceptsLanguage(language):` **CHECKS LANGUAGE**
- `request.acceptsCharset(charset):` **CHECKS**

REQUEST HEADER SHORTCUTS

- `request.ips`: **IP ADDRESSES (WITH TRUST-PROXY ON)**
 - `request.path`: **URL PATH**
- `request.host`: **HOST WITHOUT PORT NUMBER**
 - `request.fresh`: **CHECKS FRESHNESS**
 - `request.stale`: **CHECKS STALENESS**
- `request.xhr`: **TRUE FOR AJAX-Y REQUESTS**

REQUEST HEADER SHORTCUTS

- `request.protocol`: RETURNS HTTP PROTOCOL
- `request.secure`: CHECKS IF PROTOCOL IS `https`
- `request.subdomains`: ARRAY OF SUBDOMAINS
 - `request.originalUrl`: ORIGINAL URL

HTTP RESPONSES

THE RESPONSE OBJECT IS ALSO ACCESSIBLE VIA ROUTING HANDLERS IN EXPRESS

IT IS THE SECOND ARGUMENT IN THE HANDLER'S CALLBACK

```
app.get('/users/:id', function (req, res) {  
  // 'res' is the response object  
});
```

THE RESPONSE OBJECT CAN BE USED TO MODIFY AN HTTP RESPONSE BEFORE SENDING IT OUT

EXPRESS RESPONSE METHOD

- › `response.redirect(status, url):` **REDIRECT REQUEST**
- › `response.send(status, data):` **SEND RESPONSE**
- › `response.json(status, data):` **SEND JSON AND FORCE PROPER HEADERS**
- › `response.sendFile(path, options,`

HTTP STATUS CODES

TO SPECIFY A STATUS CODE, USE THE RESPONSE OBJECT'S
STATUS FUNCTION

```
app.get('/user/:id', function (req, res) {  
  // logic to check for user  
  if (!exists) {  
    res.status(404);  
  } else if (authorized) {  
    res.status(200);  
  } else {  
    res.status(401);  
  }  
  // ...  
});
```

HTTP STATUS CODES

- **2XX: FOR SUCCESSFULLY PROCESSED REQUESTS**
- **3XX: FOR REDIRECTIONS OR CACHE INFORMATION**
 - **4XX: FOR CLIENT-SIDE ERRORS**
 - **5XX: FOR SERVER-SIDE ERRORS**

NOTE: FOR 3XX STATUS CODES, THE CLIENT MUST TAKE ADDITIONAL ACTION FOLLOWING THE COMPLETION OF THE

SENDING A RESPONSE

USE THE RESPONSE OBJECT'S **SEND** FUNCTION TO SEND THE CLIENT A RESPONSE

```
app.get('...', function (req, res) {  
  res.send('Hello World!');  
});
```

SENDING A RESPONSE

THE CONTENT-TYPE IS DETERMINED GIVEN THE TYPE OF ARGUMENT PASSED

```
res.send('Hello World!');           // Content-type: text/plain
res.send([ 5, 7, 9 ]);               // Content-type: application/json
res.send({ name: 'John Doe' });     // Content-type: application/json
```

SENDING A RESPONSE

THE CONTENT-TYPE CAN ALSO BE HARDCODED

```
res.set('Content-Type', 'text/plain');  
res.send('Just regular text, no html expected!');
```

SENDING AN EMPTY RESPONSE

```
res.status(404).end();
```

SESSIONS

HTTP IS A STATELESS PROTOCOL – INFORMATION ABOUT A CLIENT IS NOT RETAINED OVER SUBSEQUENT REQUESTS

USE SESSIONS TO OVERCOME THIS PROBLEM

ENABLE THE `cookieParser` AND `session` MIDDLEWARES TO PROCESS COOKIES

SESSIONS

```
app.use(express.cookiesParser());  
app.use(express.session({ secret: 'notastrongsecret' }));
```

THE SESSION IS NOW ACCESSIBLE VIA `request.session`

```
app.get('...', function (req, res) {  
  var session = req.session;  
});
```

REDIS STORE WITH EXPRESS

```
$ npm install connect-redis express-session
```

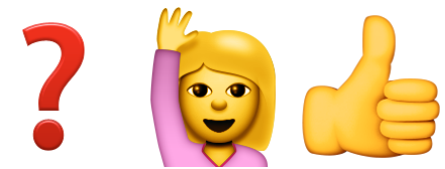
```
var session = require('express-session'),  
    RedisStore = require('connect-redis')(session);
```

```
app.use(session({  
  store: new RedisStore(options),  
  secret: 'keyboard cat'  
}));
```

LOAD-BALANCING

- **CLUSTERS**
 - **NGINX**
 - **HAPROXY**
 - **VARNISH**

QUESTIONS AND EXERCISES



WORKSHOP



```
$ npm i -g expressworks
```

```
$ npm i -g meanworks
```