Back-end Web Development

Introduction

Some words about the **OSI Model**

7 - Application

6 - Presentation

5 - Session

4 - Transport

3 - Network

2 - Data link

1 - Physical

HTTP

> The Hypertext Transfer Protocol (HTTP) is a **stateless** application-level protocol for distributed, collaborative, hypertext information systems.

1989 - First proposal of Tim Berners-Lee at CERN (HTTP + HTML)

1991 - HTTP/0.9

1996 - tools.ietf.org/html/rfc1945 (HTTP/1.0)

1997 - ~~tools.ietf.org/html/rfc2068 (obsolete)~~

2014 - **tools.ietf.org/html/rfc7230** (HTTP/1.1)

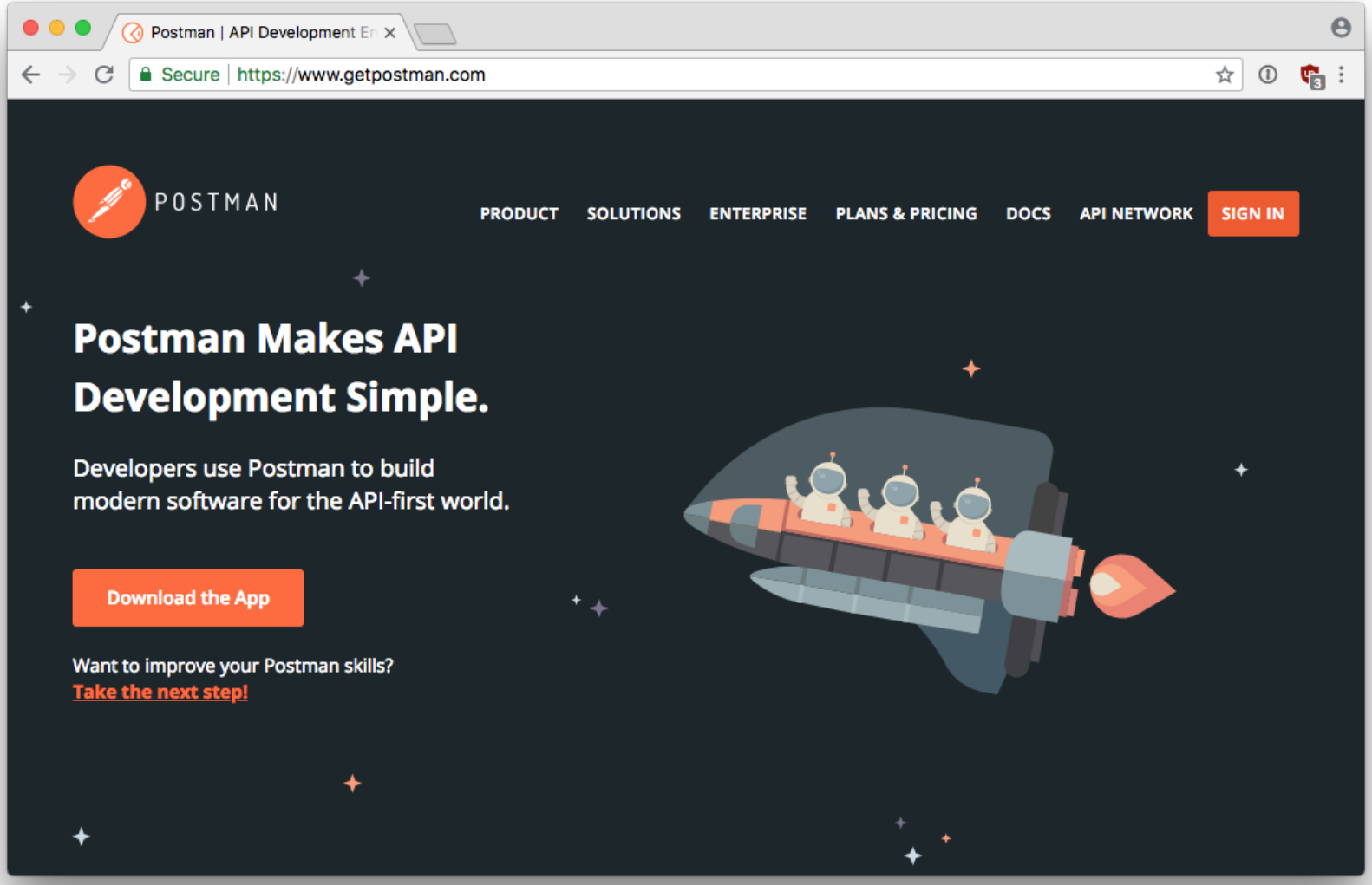
2015 - tools.ietf.org/html/rfc7540 (HTTP/2)

> Derived from SPDY, Google experiment (2012)

A client/server protocol

```
telnet
```

```
curl
```


Postman | API Development Er
Secure | https://www.getpostman.com
POSTMAN
PRODUCT
SOLUTIONS
ENTERPRISE
PLANS & PRICING
DOCS
API NETWORK
SIGN IN
Postman Makes API Development Simple.
Developers use Postman to build modern software for the API-first world.
Download the App
Want to improve your Postman skills?
Take the next step!

Anatomy of an HTTP request

- URL (Path)
- Method
- Headers
- (Body)

Methods

- `GET`
- `POST`
- `PATCH`
- `DELETE`

Request Headers

- `Authorization`
- `Accept-Language`
- `Content-Type`
- `Cookie`
- `Referer`
- `User-Agent`

Response Headers

- `Cache-Control`
- `Content-Type`
- `Location`
- `Set-Cookie`
- `Status`

Response Status codes

- `2xx` - Here you go
- `3xx` - Go away
- `4xx` - You fucked up *(client)*
- `5xx` - I fucked up *(server)*

Credit to @stevelosh

`2xx` Success

```
200 OK
201 Created
204 No Content
```

`3xx` Redirection

```
301 Moved Permanently
302 Found
304 Not Modified
```

`4xx` Client errors

```
403 Forbidden
404 Not Found
422 Unprocessable Entity
```

`5xx` Server errors

```
500 Internal Server Error
502 Bad Gateway
504 Gateway Timeout
```

REST

Representational State Transfer

> An architectural style that defines a set of constraints to be used for creating web services.

RESTful web services allow the requesting systems to access and manipulate textual representations of web resources by using a **uniform and predefined set of stateless operations**.

Resource

Originally: file identified by a URL on th www.

In REST: key abstraction of information.

Representation: e.g. JSON of a database row

REST Interface

- Resource identification in requests (URI)
- Resource manipulation through representations (JSON)
- Self-descriptive messages (Content-Type)
- HATEOAS: Hypermedia as the engine of application state

Exemple: api.github.com

CRUD on a Resource

Resource exemple: a `product`

Verb	Action	Exemple
GET	Retrieve a list of resources	`GET /products`
POST	Create a new resource	`POST /products`
GET	Retrieve a resource	`GET /products/:id`
PATCH	Update an existing resource	`PATCH /products/:id`
DELETE	Delete a resource	`DELETE /products/:id`

Flask

A micro web framework for Python

License: **BSD**

What Flask does not have *(by default)*

- Database ORM
- Form Validation
- Authentication

But: Rich ecosystem of third-party `flask-*` libraries

flask.pocoo.org/docs/1.0/foreword

Set-up

```
mkdir flask-101 && cd $_
pipenv --python 3.7
pipenv install flask
touch wsgi.py
```

Minimal Flask application

```python
# wsgi.py
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello World!"
```

Start the development server with:

```
FLASK_ENV=development pipenv run flask run
```

Testing - Setup

```
mkdir tests
touch tests/test_wsgi.py
pipenv install flask-testing nose --dev
```

Testing - My first Flask test

```
# test/test_wsgi.py
from flask_testing import TestCase
from wsgi import app

class ApplicationTest(TestCase):
    def create_app(self):
        app.config['TESTING'] = True
        return app

    def test_home(self):
        response = self.client.get("/")
        body = response.data.decode()
        self.assertEqual(response.status_code, 200)
        self.assertTrue("Hello" in body)
```

Happy REST-ing!