Web development - Intermediate

Agenda

Day 1(How the web works):

- How the web works
- Client-server architecture
- Evolution of web(WWW vs Internet)
- Set up Developer Environment

Day 2(Bootstrap, JQuery and DOM manipulation):

- Bootstrap
- DOM
- DOM selectors and events
- jQuery

Day 3(HTTP/JSON/AJAX +Async JS):

- HTTP/HTTPS
- JSON
- AJAX
- Asynchronous JavaScript

Agenda

Day 4-5(Frameworks, React):

- Introduction about frameworks and How the frameworks work under the hood?
- Introduction to React(state, props, component)

Day 6(APIs and microservices):

- How APIs work?
- Evolution of APIs
- Micro services and Web services

Day 7(Backend):

- Basics
- Introduction to NodeJs and ExpressJs

Day 6

- React Todolist and React routing example
- How APIs work?
- Evolution of APIs
- Micro services and Web services

React Routing

Resources :

https://bit.ly/2Ra9Kw2

https://reactrouter.com/web/example/basic

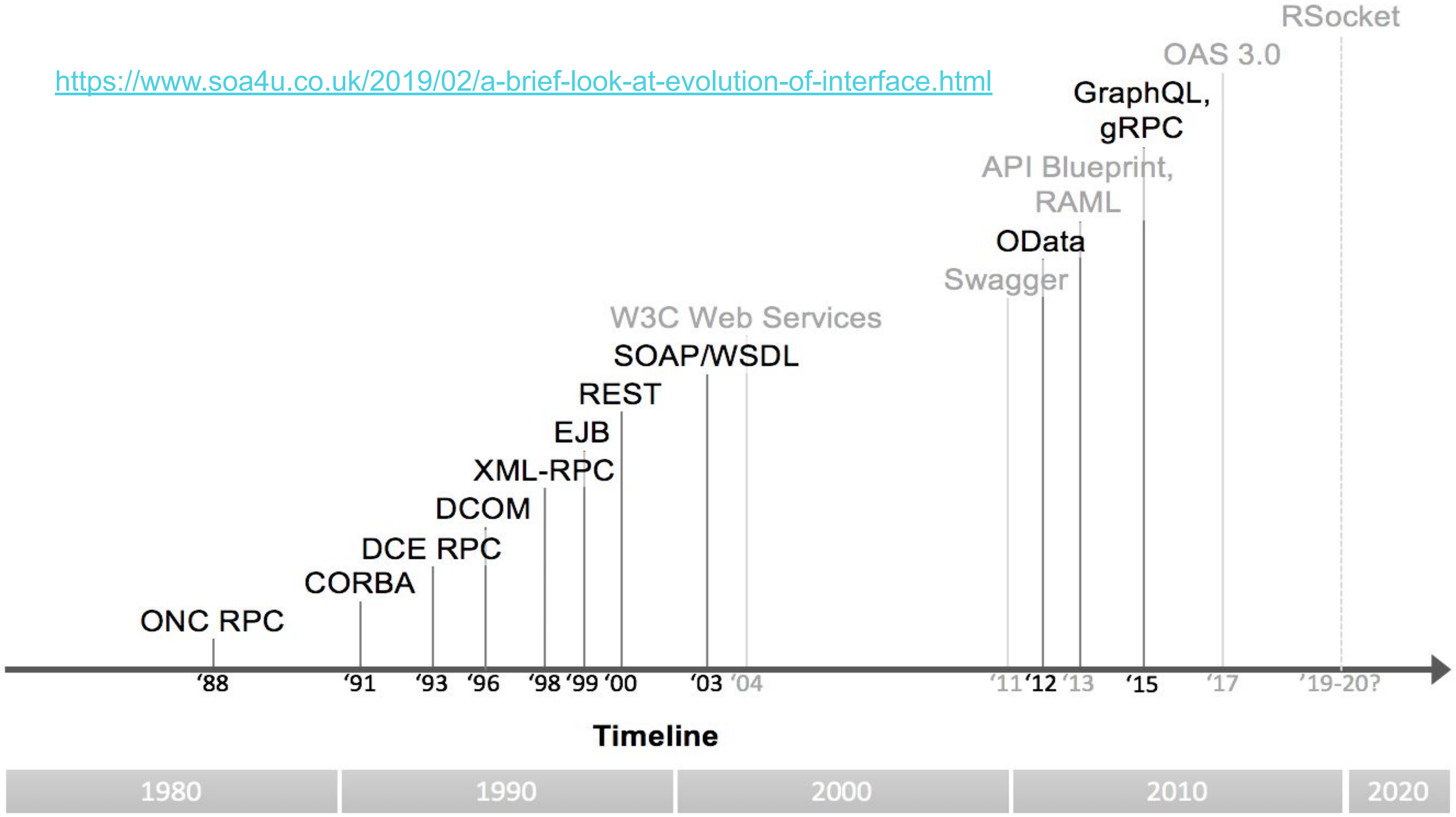
API

- software intermediary that allows two applications to talk to each other
- simplify and speed up software development.
- serve as an abstraction layer between two systems, hiding the complexity and working details

Resources:

https://blogs.mulesoft.com/learn-apis/api-led-connectivity/what-are-apis-how-do-apis-work/

https://www.soa4u.co.uk/2019/02/a-brief-look-at-evolution-of-interface.html

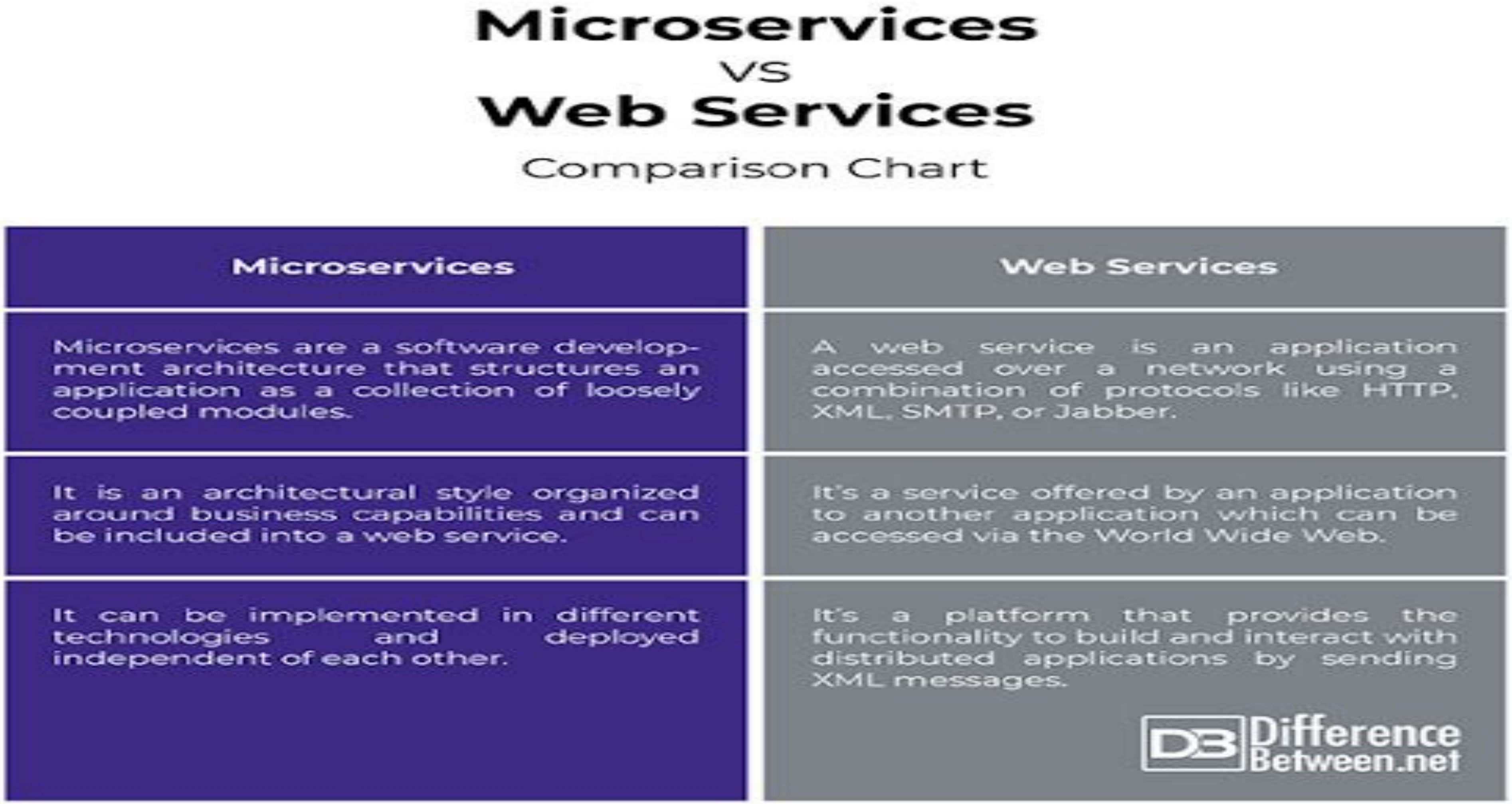


Benefits of REST APIs

- Very easy to learn and understand;
- It provides developers with the ability to organize complicated applications into simple resources;
- It easy for external clients to build on your REST API without any complications;
- It is very easy to scale;
- A REST API is not language or platform-specific, but can be consumed with any language or run on any platform.

Try consuming a REST API with React:

https://www.smashingmagazine.com/2020/06/rest-api-react-fetch-axios/



Microservices VS Web Services

Comparison Chart

Microservices	Web Services
Microservices are a software development architecture that structures an application as a collection of loosely coupled modules.	A web service is an application accessed over a network using a combination of protocols like HTTP, XML, SMTP, or Jabber.
It is an architectural style organized around business capabilities and can be included into a web service.	It's a service offered by an application to another application which can be accessed via the World Wide Web.
It can be implemented in different technologies and deployed independent of each other.	It's a platform that provides the functionality to build and interact with distributed applications by sending XML messages.

SERVICE ORIENTED ARCHITECTURE	MICROSERVICES ARCHITECTURE
Maximizes application service reusability	Focused on decoupling
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps and Continuous Delivery are becoming popular, but are not mainstream	Strong focus on DevOps and Continuous Delivery
Focused on business functionality reuse	More importance on the concept of “bounded context”
For communication it uses Enterprise Service Bus (ESB)	For communication uses less elaborate and simple messaging systems
Supports multiple message protocols	Uses lightweight protocols such as HTTP, REST or Thrift APIs
Use of a common platform for all services deployed to it	Application Servers are not really used, it’s common to use cloud platforms
Use of containers (such as Docker) is less popular	Use of containers (such as Docker) is less popular
SOA services share the data storage	Each microservice can have an independent data storage
Common governance and standards	Relaxed governance, with greater focus on teams collaboration and freedom of choice

Thank you!