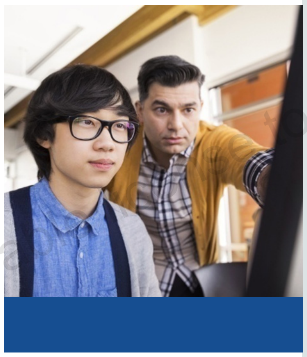
13

HTTP and REST Fundamentals

Objectives

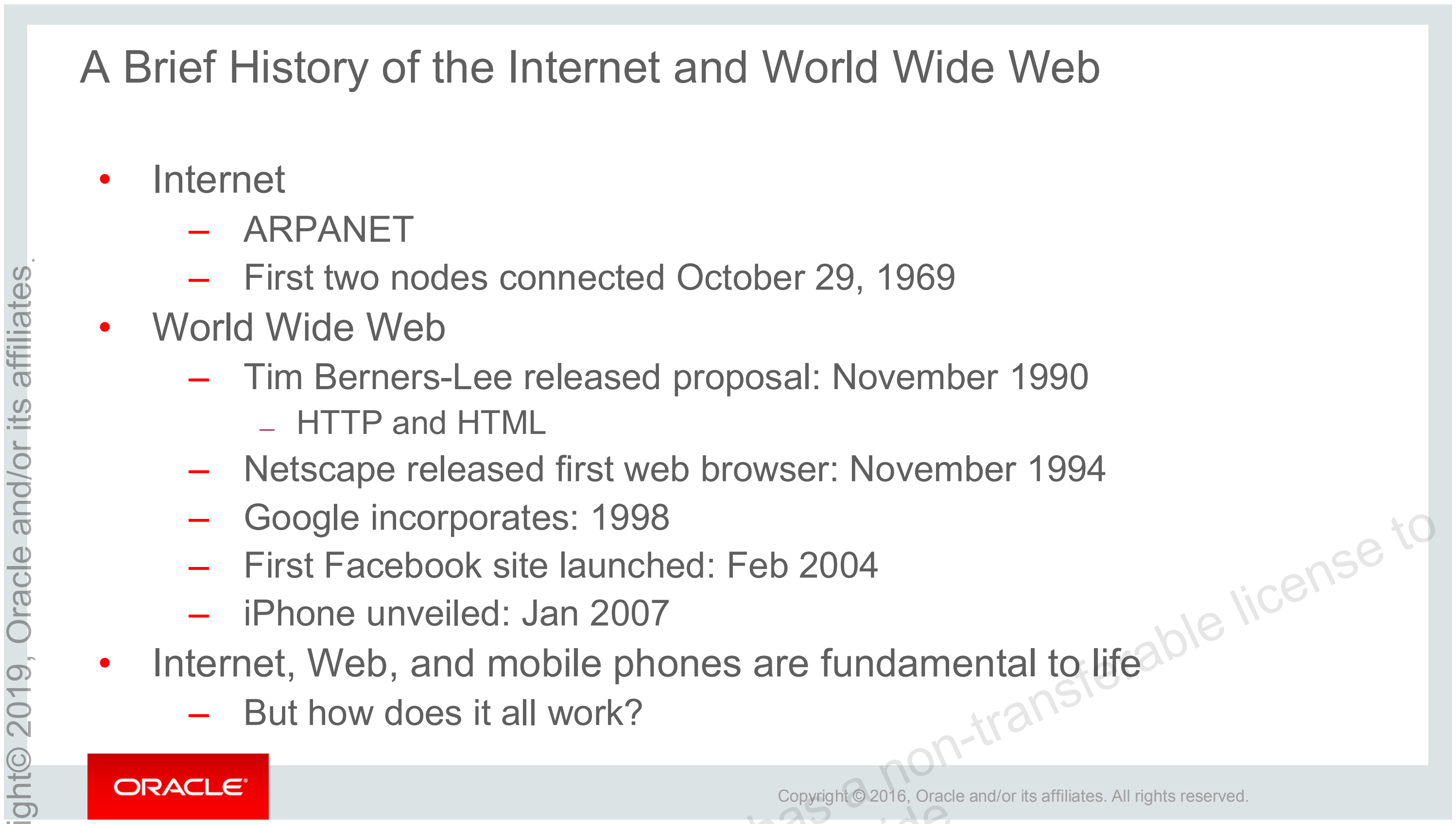
After completing this lesson, you should be able to:

- Define HTTP
- Describe the steps in an HTTP transaction
- List the key HTTP headers in request and response
- List the HTTP Methods
- Define the structure of a URL and a URI
- Describe a REST Web Service
- Describe the JSON format



Topics

- **The Internet and World Wide Web**
- HTTP (HyperText Transfer Protocol)
- REST: Introduction
- REST in Action



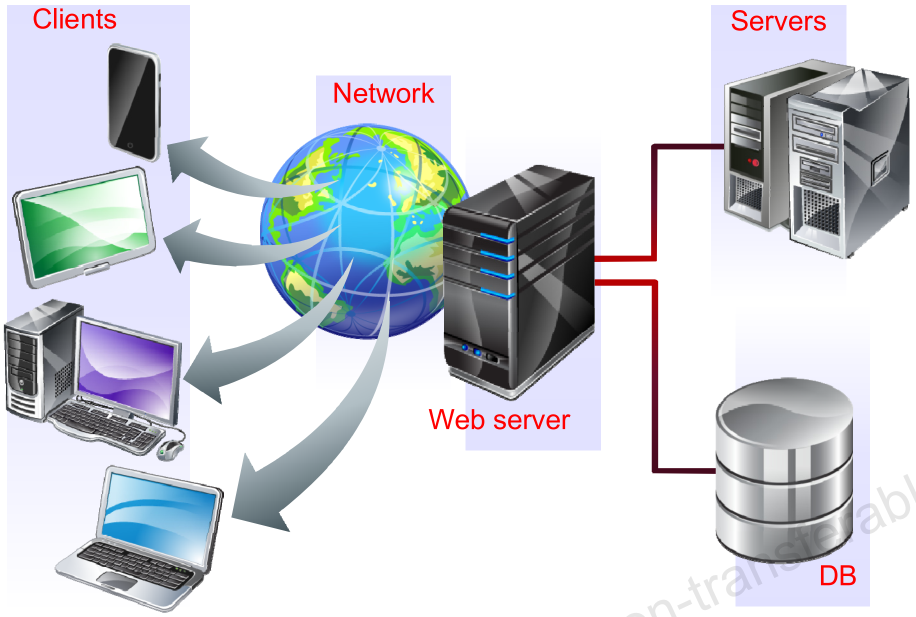
A Brief History of the Internet and World Wide Web

- Internet
  - ARPANET
  - First two nodes connected October 29, 1969
- World Wide Web
  - Tim Berners-Lee released proposal: November 1990
    - HTTP and HTML
  - Netscape released first web browser: November 1994
  - Google incorporates: 1998
  - First Facebook site launched: Feb 2004
  - iPhone unveiled: Jan 2007
- Internet, Web, and mobile phones are fundamental to life
  - But how does it all work?

- First two nodes of what would become the ARPANET were interconnected between UCLA (Los Angeles) and SRI International in Menlo Park, CA on 29 October 1969.
- Tim Berners-Lee published a formal proposal for HTML and HTTP in November 1990. He proposed building a "Hypertext project" he called "WorldWideWeb" as a "web" of "hypertext documents" to be viewed by "browsers" using a client–server architecture.
  - **HTTP:** HyperText Transfer Protocol
  - **HTML:** HyperText Markup Language
- Netscape released the first commercial version of its browser "Mosaic Netscape 0.9" on October 13, 1994.
- Mark Zuckerberg launched his first version of "thefacebook" on Februrary 4, 2004.
- iPhone unveiled by Steve Jobs at MacWorld January 9, 2007.

Note: All information referenced from Wikipedia

How Does It All Work?



On the web, clients connect to servers by using a network, commonly the Internet. Servers can be connected with one another to request information to complete requests from clients.

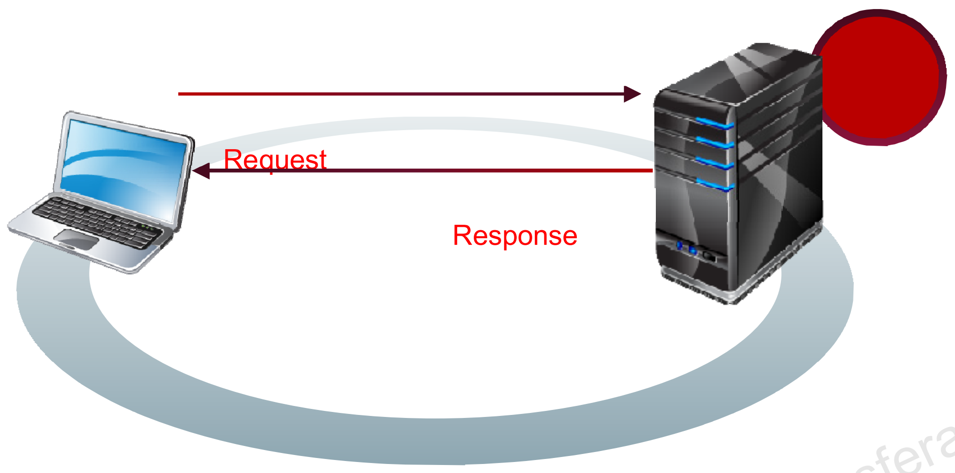
What Are Web Servers?

Web applications are usually stored in web servers. They:

- Handle web requests
- Store application files
- Provide access to resources, such as:
  - JSON data
  - HTML pages
  - Dynamic pages
  - Images
  - Video

- A web server is software that listens for requests, provides access to resources, and generates a response.
- A web server can receive requests from multiple clients, and handles them separately.
- When a request is received, the server locates the resource, and sends its contents back to the client. Typically, the response is an HTML file, which the browser displays. In some cases, requests are forwarded to other services that construct the content of the response.
- Dynamic content can be generated in the server or in the client. Web applications use dynamic content in the client with JavaScript and in the server by using an application server.

How Do Web Servers Work?



Web servers work in three stages:

1. A request is always initiated by the client.
2. The server locates the resource and constructs a response.
3. The response with the content of the resource is sent back to the client.

Note that these three steps can be repeated multiple times.

What Is a Client?

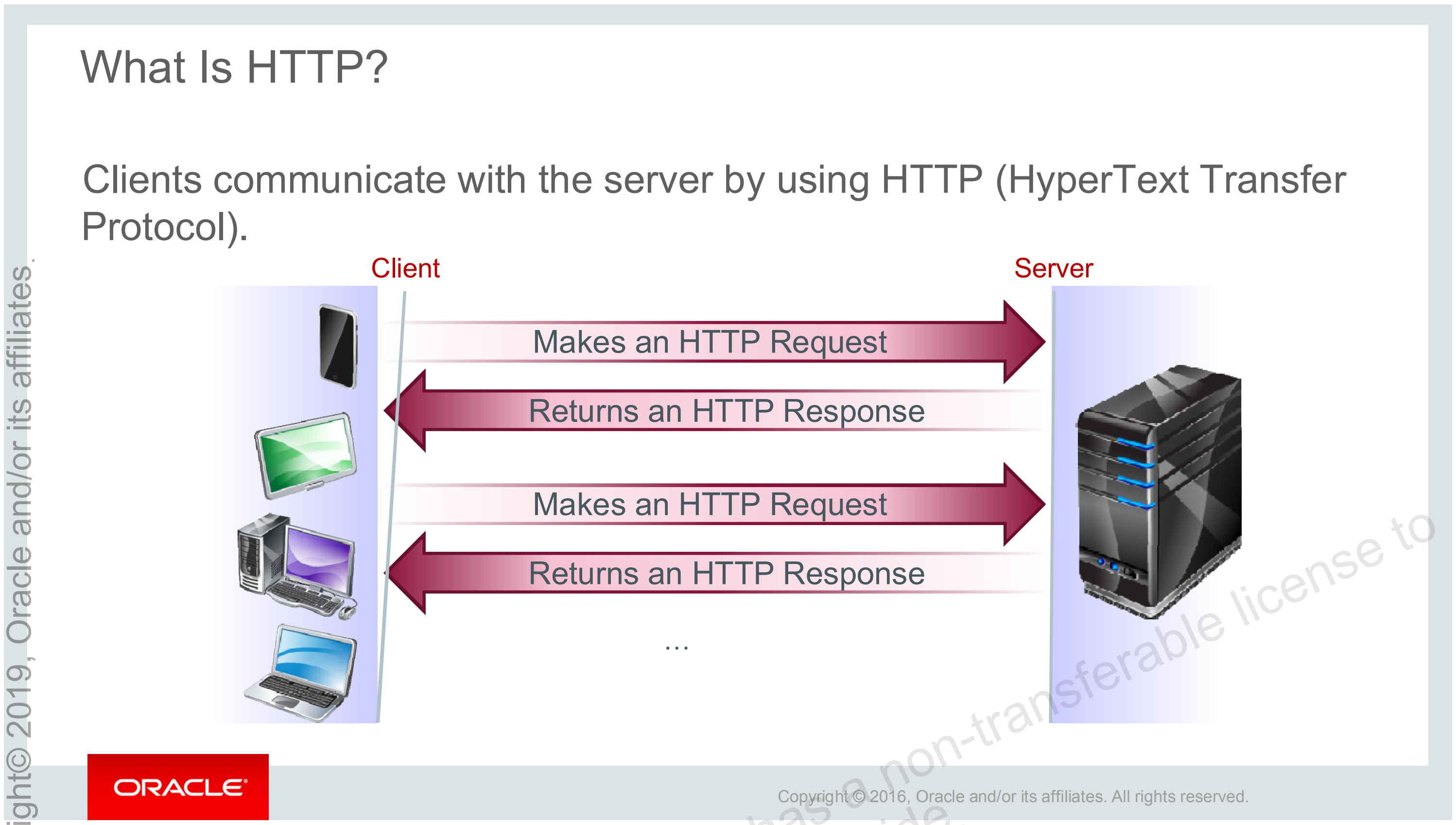
A client is an application that runs on a machine that can make requests from a web server and HTML response data,

- **Web browsers**
  - Firefox, Chrome, Internet Explorer, and Safari
- Other applications
  - Weather app on mobiles
  - News readers

In this course, client applications are created using HTML pages run in your computer.

Topics

- The Internet and World Wide Web
- **HTTP (HyperText Transfer Protocol)**
- REST: Introduction
- REST in Action

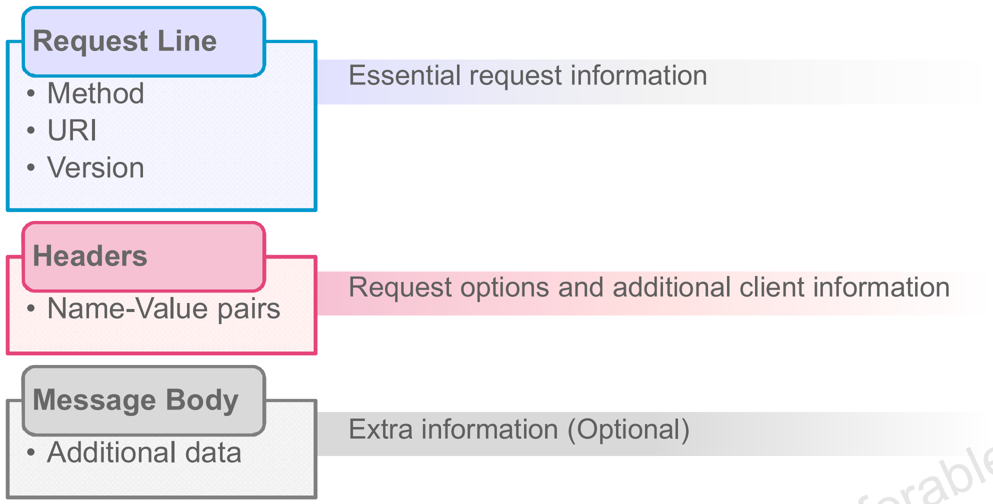


What Is HTTP?

Clients communicate with the server by using HTTP (HyperText Transfer Protocol).

- HTTP is a communication protocol, which relies on TCP connections over IP to connect the client and the server.
- In the HTTP protocol, all requests are independent.
- The client might trigger additional requests, but each one of these requests is completely new to the server and is handled independent of previous requests.
- There is no session management as part of the HTTP protocol.
- HTTP is stateless.
- Session management is accomplished by sending an identifier as part of the request that can relate to a specific session.

HTTP Request


Request Line
• Method
• URI
• Version
Essential request information
Headers
• Name-Value pairs
Request options and additional client information
Message Body
• Additional data
Extra information (Optional)

A request contains some information that helps the server generate a response.

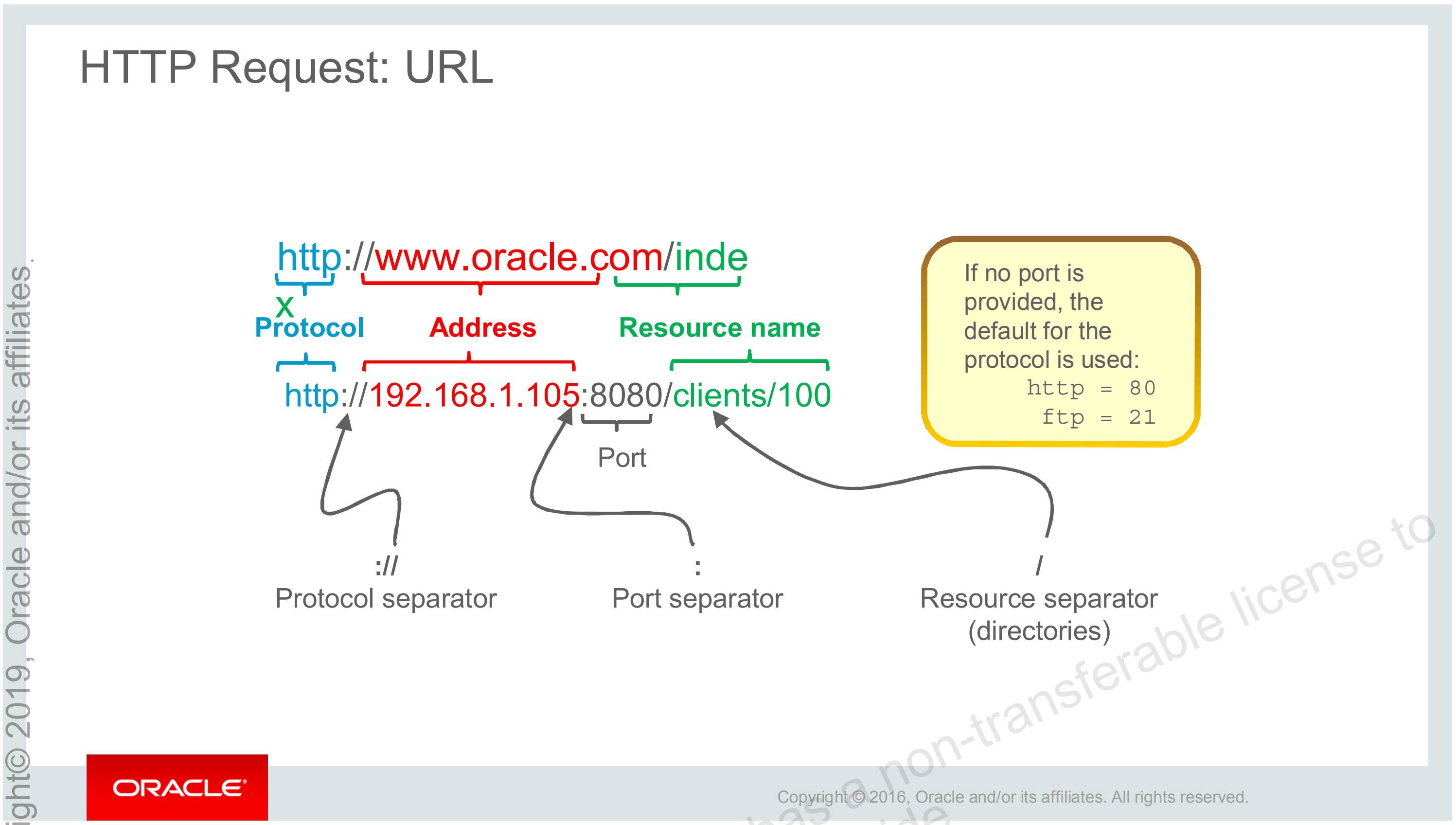
Every request contains a Request Line. This contains the Uniform Resource Identifier (URI), which is the name of the requested resource, and the Method used. HTTP defines the following methods: `OPTIONS`, `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `TRACE`, and `CONNECT`. Web browsers use only `GET` or `POST`.

The HTTP request headers are key-value pairs that are used to provide additional information.

- `accept-encoding`: Defines what encoding is expected in the response
- `user-agent`: Specifies the client application being used. When you are using a browser, the name and version of the browser are supplied in this header.
- Session IDs and session information, such as cookies, can be contained in request headers.

For more information about HTTP headers, refer to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.

The Message Body can contain additional information supplied by the user, such as form data when using the POST method.



HTTP Request: URL

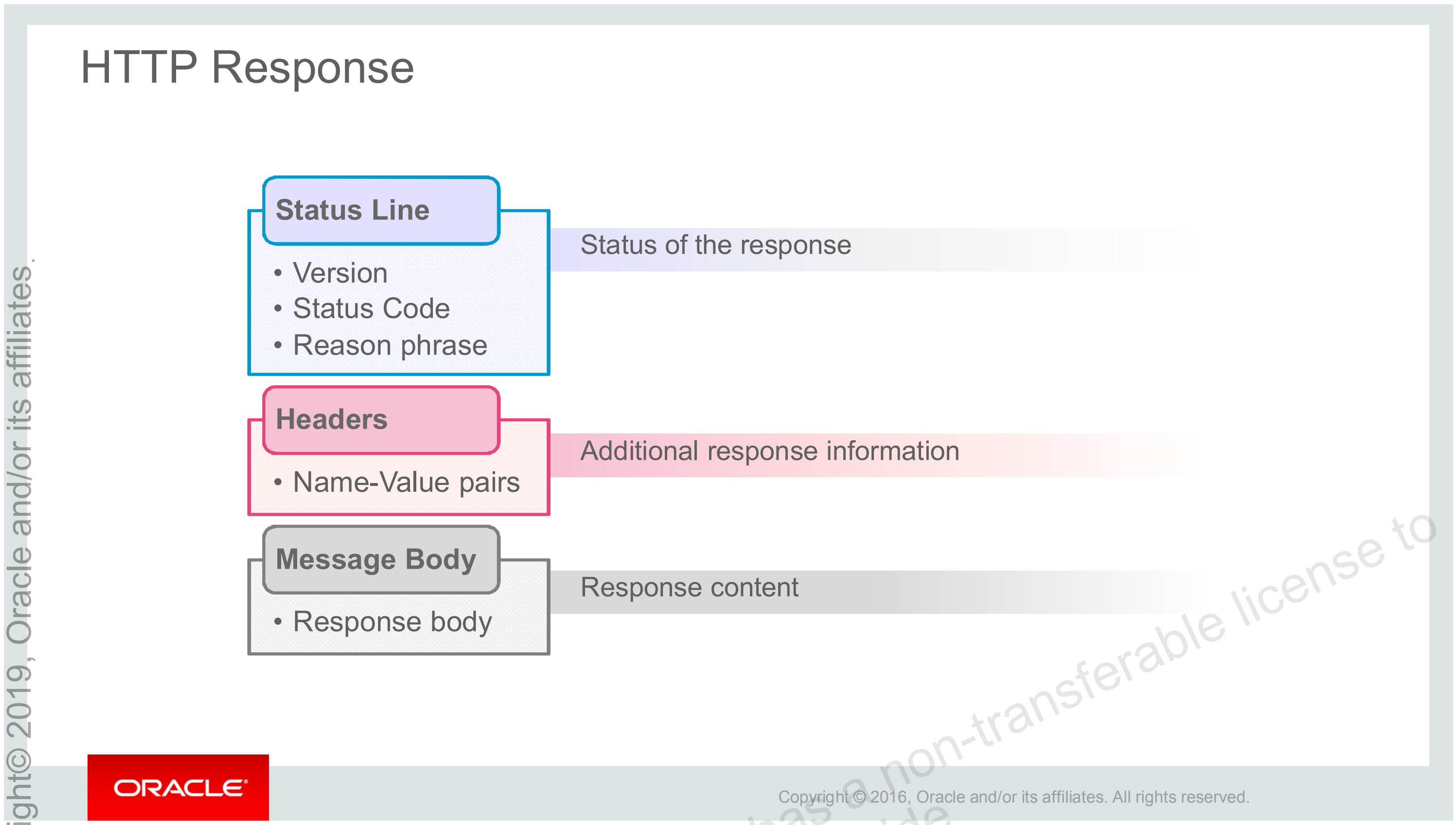
http://www.oracle.com/index
Protocol
Address
Resource name
http://192.168.1.105:8080/clients/100
Port
://
Protocol separator
:
Port separator
/
Resource separator (directories)

If no port is provided, the default for the protocol is used:

```
http = 80
 ftp = 21
```

The slide shows the different components that compose a URL.

Remember that if no port is specified, the default is used. For HTTP, the default port is 80; for FTP, it is 21.



HTTP Response

Status Line
- Version
- Status Code
- Reason phrase

Status of the response

Headers
- Name-Value pairs

Additional response information

Message Body
- Response body

Response content

ORACLE

The HTTP response must contain the status line.

Status codes are grouped in the following ways:

- **1xx:** Informational, for example, 101 Switching Protocols
- **2xx:** Success, for example, 200 OK
- **3xx:** Redirection. Further action is required to complete the request, for example, 301 Moved Permanently.
- **4xx:** Client Error. The request contains a problem and could not be fulfilled, for example, 404 Not Found.
- **5xx:** Server Error. There was a problem in the server, for example, 500 Server Error.

You can find the list of response codes at http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

Response headers contain additional information about the response, such as the content type, content size, user cookies, session ID, and so on.

Between the headers and the body, there needs to be an empty line.

The body contains the actual content of the requested resource. The response content is optional.

Response Bodies

A response body contains the content of a resource, including:

- JSON data *
- Documents
- Images
- Audio
- Video
- JavaScript files

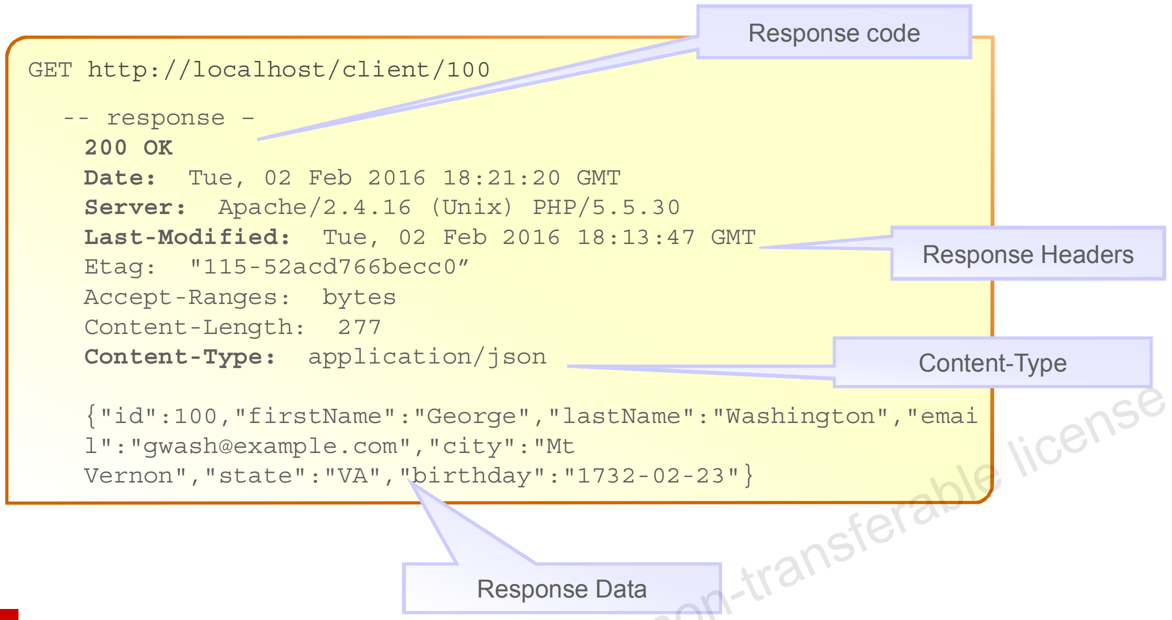
In summary, the response body contains the requested data.

The response body contains the content of the requested resource.

The client uses the response content to display information on the screen.

*JSON is a form of representing data as JavaScript literals.

An HTTP Response



```
GET http://localhost/client/100

   -- response –
    200 OK
    Date:  Tue, 02 Feb 2016 18:21:20 GMT
    Server:  Apache/2.4.16 (Unix) PHP/5.5.30
    Last-Modified:  Tue, 02 Feb 2016 18:13:47 GMT
    Etag:  "115-52acd766becc0"
    Accept-Ranges:  bytes
    Content-Length:  277
    Content-Type:  application/json

    {"id":100,"firstName":"George","lastName":"Washington","emai
    l":"gwash@example.com","city":"Mt
    Vernon","state":"VA","birthday":"1732-02-23"}
```

An HTTP response includes header information along with the body.

- The body includes JSON data.
- Notice that a response code of 200 is returned, which indicates that the HTTP transaction returned OK.
- The Content-Type header indicates what is included in the HTTP response.

Topics

- The Internet and World Wide Web
- HTTP (HyperText Transfer Protocol)
- **REST: Introduction**
- REST in Action

What Is REST?

REST is the architecture that the entire web uses. It is based on the following design principles:

- Client/server interactions
- Uniform interface
- Layered system
- Cache
- Stateless
- Code-on-demand

Representational State Transfer (REST) is a term that has existed for a long time. Roy Fielding describes REST in his doctoral dissertation at UC Irvine, and defines the web architecture for servicing multiple clients in a flexible, scalable, and uniform way.

REST takes full advantage of the HTTP protocol. The complete HTTP protocol complies entirely with the REST proposed architecture:

- Multiple and diverse clients exist in the form of web browsers and applications.
- Uniform Resource Locators and HTTP methods are present to access and modify resources.
- It can be a system that may contain many layers in between the client and the server.
- Caches can be implemented on resources because often the same operations performed over the same resources produce the same result.
- Stateless operations establish no sequence or dependency in REST calls.
- Finally, servers may provide code that needs to be executed. Most often, instead of providing code, the server provides further possible operations in the form of HyperMedia in such a way that clients know what REST operations are possible.

How Do I Use REST on the Web?

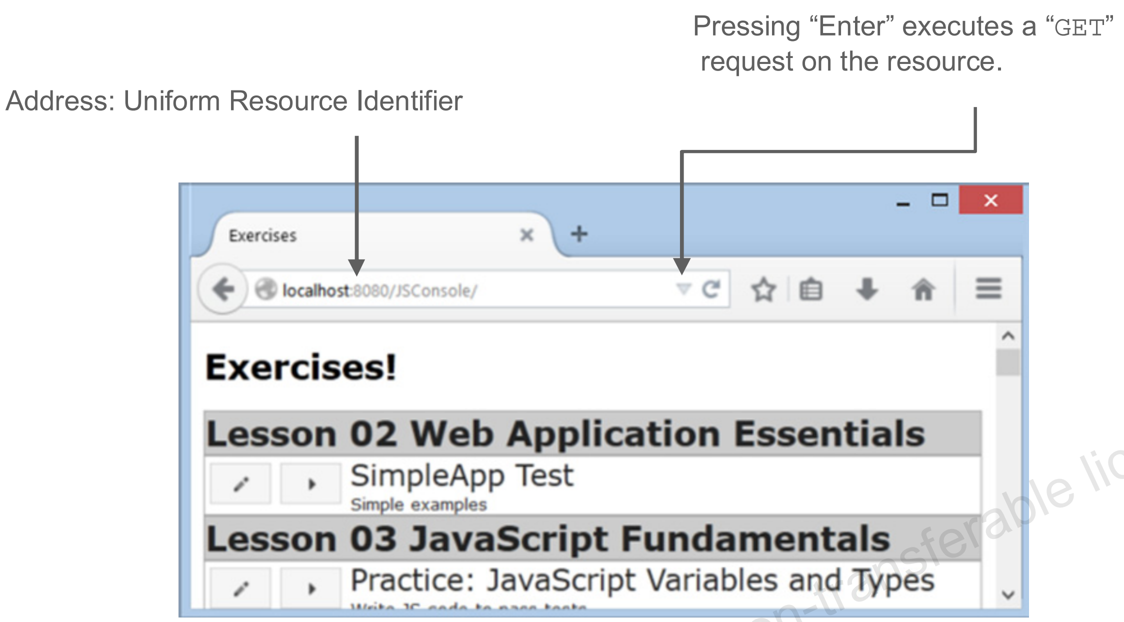
You are already using REST.

- Client/server interactions: Browser – Web Server
- Uniform interface: HTTP, URL
- Layered System: IP: Proxies, Gateways, and so on
- Cache: Web browser cache, intermediary caches
- Stateless: HTTP request-and-response model
- Code-on-demand: JavaScript code

The URI and Method represent the resource to be accessed or modified.

Use the request headers to provide additional information about the request. If needed, add a `requestBody` to provide data in the request.

Practical REST


Pressing "Enter" executes a "GET" request on the resource.
Address: Uniform Resource Identifier
Exercises
localhost:8080/JSConsole/

Exercises!

Lesson 02 Web Application Essentials

SimpleApp Test

Simple examples

Lesson 03 JavaScript Fundamentals

Practice: JavaScript Variables and Types

In the example in the slide, REST begins in the web browser when you enter an address and press Enter.

The browser makes an HTTP `GET` request to get the resource from a server that is specified by the Uniform Resource Identifier provided in the address bar.

Uniform Resource Identifier or Uniform Resource Locator

- A URI is composed of the name of the resource, but does not specify the way to access such resource.
- A URL has the protocol or way to get a particular resource.

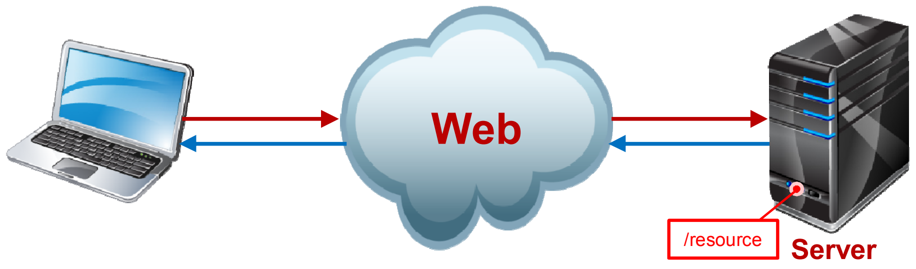
In the end, all URLs are also URIs but the term URI is more widely accepted when referring to addresses because sometimes the protocol is omitted, for example:

- http://www.oracle.com/education is a valid URL and URI.
- www.oracle.com/education is a valid URI only.

Thus, using URI for both is correct and more "universal."

Practical REST

GET : http://server/resource



The request is sent through the network to the server, which locates the resources and applies the method. In the example in the slide, the method is a `GET` request. Thus, the server will create a representation of the resource and send it back to the client.

Can I Use REST from the Command Line?

- Use cURL:
  - Is an open source command-line tool
  - Enables user authentication, secure connections

Example:

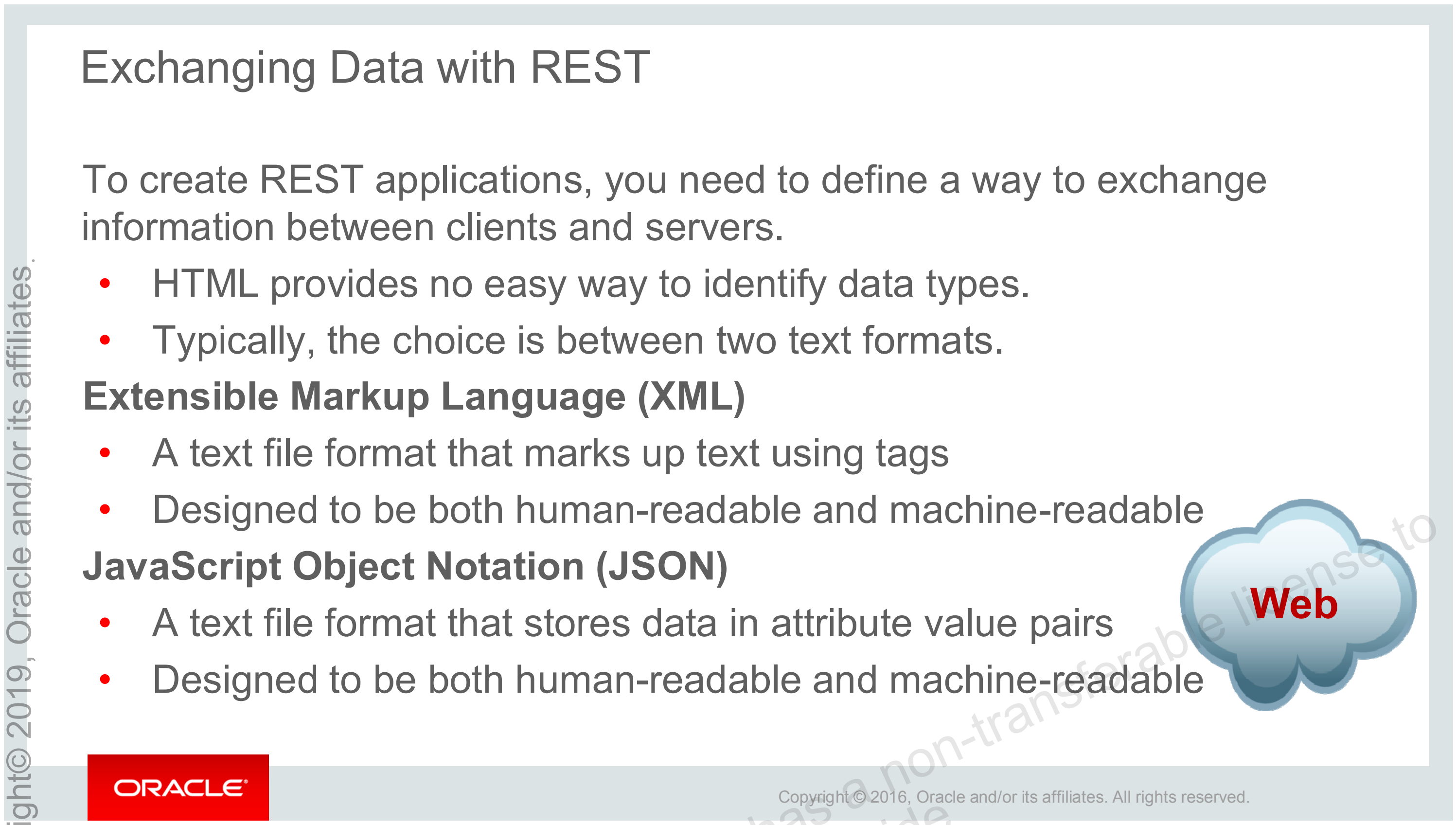
```
curl -v -X PUT \ -H "X-Auth-Token: AUTH_tkb4fdf39c92e9f62cca9b7c196f8b6e6b"
     \ https://foo.storage.oraclecloud.com/v1/Storage-
     myIdentity3/FirstContainer
```

cURL is an open source, command-line tool for transferring data with URL syntax, supporting various protocols including HTTP and HTTPS.

In the example in the slide, cURL is used to create a container, FirstContainer, in the Oracle Storage Cloud Service, using the authentication token specified in the same command. The –v option specifies a verbose output, and the –X option specifies the method to be used, in this case `PUT`. You will learn more about HTTP methods later in this lesson.

Topics

- The Internet and World Wide Web
- HTTP (HyperText Transfer Protocol)
- REST Introduction
- **REST in Action**



Exchanging Data with REST

To create REST applications, you need to define a way to exchange information between clients and servers.

- HTML provides no easy way to identify data types.
- Typically, the choice is between two text formats.

Extensible Markup Language (XML)

- A text file format that marks up text using tags
- Designed to be both human-readable and machine-readable

JavaScript Object Notation (JSON)

- A text file format that stores data in attribute value pairs
- Designed to be both human-readable and machine-readable

ORACLE

Extensible Markup Language (XML)

A text file format that marks up text using tags. For example:

```
<root>
     <name>John Doe</name>
     <email>jdoe@example.com</email>
</root>
```

JavaScript Object Notation (JSON)

A text file format that stores data in attribute value pairs

```
  {"name":"John Doe", "email":"jdoe@example.com"}
```

Both formats are designed to be human-readable and machine-readable.

XML Versus JSON in Web Services

Both formats have advantages and disadvantages.

- **Extensible Markup Language (XML)**
  - A large number of processing libraries exist to support XML in almost every language (SAX, DOM, StAX, and JAXB).
  - Representation of data is verbose.
  - May require additional system resources to parse and process
- **JavaScript Object Notation (JSON)**
  - A subset of JavaScript
  - Less verbose than XML
  - More lightweight to transport and process
  - Support evolving in other languages

JSON has become a much more popular option given its compactness and ease to process.

What Is JSON?

JavaScript Object Notation is a simple format to represent objects, arrays, and values as strings.

JavaScript includes functions to convert objects, arrays, and values to JSON strings and vice versa.

http://json.org/

ORACLE

The JavaScript Object Notation (JSON) is a format to represent JavaScript objects, arrays, and values as strings. It is one of the preferred ways of sending and receiving data using REST web services.

JSON allows you to represent objects as key-pair values and arrays as sequential lists of items.

Values can also be represented as JSON, allowing you to create all sorts of objects by using strings, numbers, and Booleans.

JSON is often compared to XML because both can be used to represent structured data. The main difference between JSON and XML is that JSON contains only data without any schema information. Therefore, JSON has no direct validation mechanism.

Also JSON contains only string, number, and Boolean value types and Object and Array data structures. You do not need to define custom data types or nodes as in XML. In JSON, data is represented as it would be inside a JavaScript object, making it extremely flexible at the exchange of validation and schema facilities.

JSON: Overview

JSON is the string (text) representation of JavaScript objects and values.

- JSON is very similar to declaring JavaScript literal values.
- JSON does not have a representation for functions.
- Only values are allowed in JSON.

ORACLE

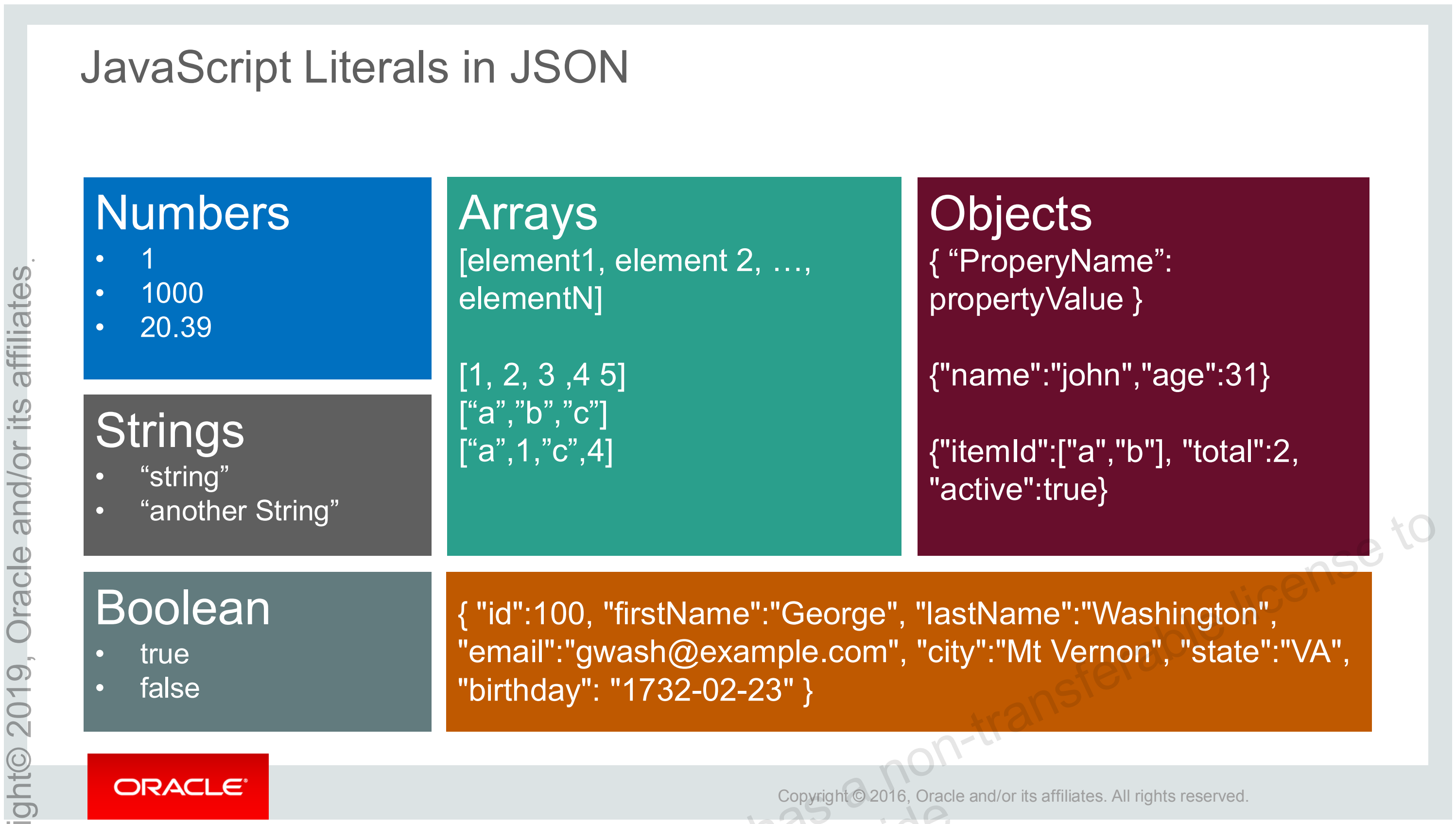
JSON values are represented in the following manner:

- **Numbers:** Use the number literal.
- **Strings:** Enclose the string literal inside double quotation marks.
- **Booleans:** These are either true or false.

JSON structures are represented in the following manner:

- **Arrays:** Must be enclosed in square brackets "[]" and elements must be comma-separated. They can contain arrays, objects, numbers, strings, or boolean as elements.
- **Objects:** Must be enclosed in curly brackets "{}." Properties are comma-separated key-value pairs. The key for the property must be a string; therefore, it must be enclosed in double quotation marks. Values for properties can be arrays, objects, numbers, strings, or boolean.

There is no way to represent functions in JSON because it is used to represent only data.



JavaScript Literals in JSON

Numbers
- 1
- 1000
- 20.39

Strings
- “string”
- “another String”

Boolean
- true
- false

Arrays

[element1, element 2, …, elementN]

[1, 2, 3 ,4 5]
[“a”,”b”,”c”]
[“a”,1,”c”,4]

Objects

{ “ProperyName”: propertyValue }

{"name":"john","age":31}

{"itemId":["a","b"], "total":2, "active":true}

{ "id":100, "firstName":"George", "lastName":"Washington", "email":"gwash@example.com", "city":"Mt Vernon", "state":"VA", "birthday": "1732-02-23" }

ORACLE

An object property value can be any other JSON literal value including even another object.

Customer example shows JSON data:

- Note that the ID is shown as a number.
- Notice pairs of fieldname/value.
- All other data pairs are strings.

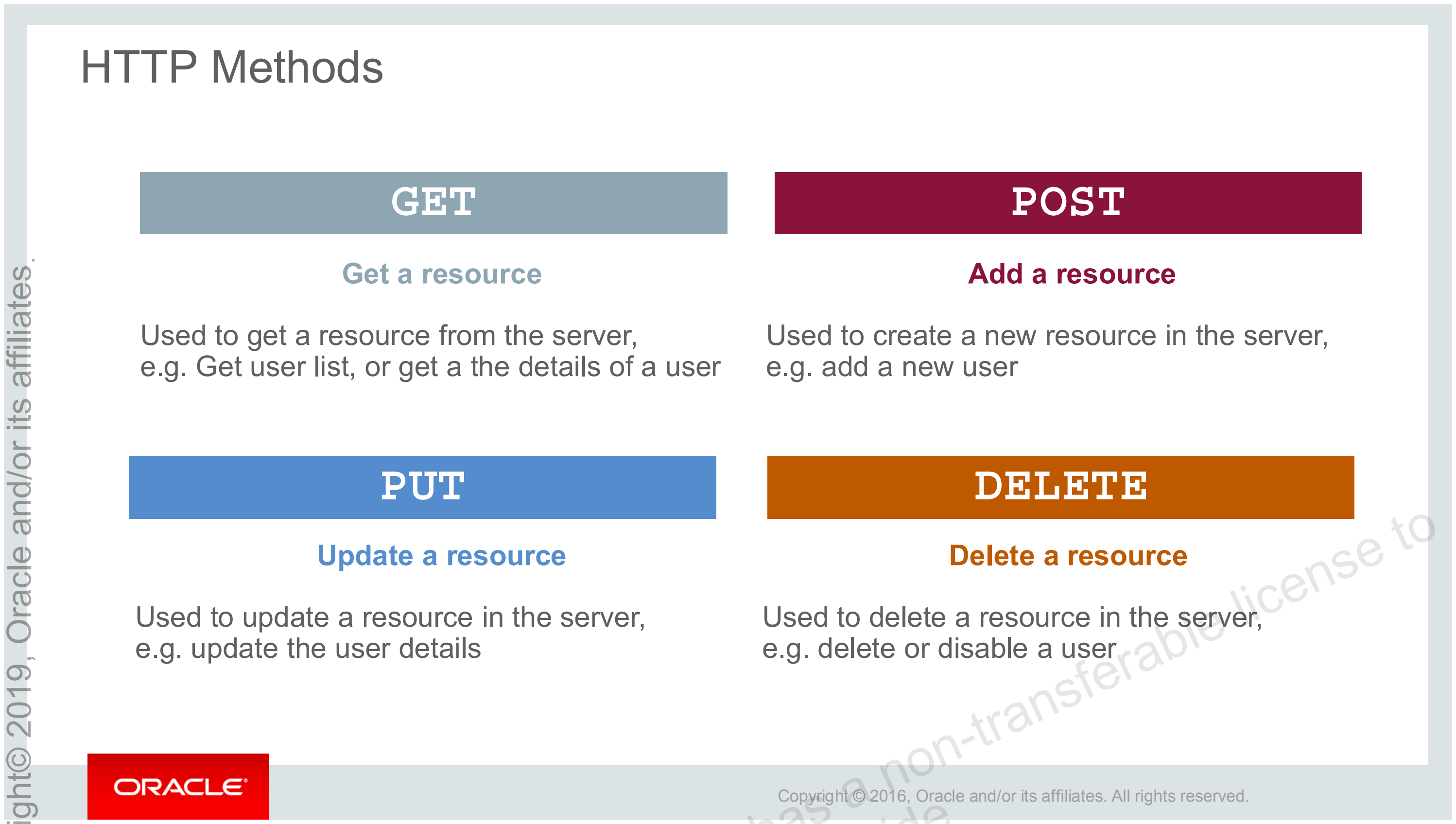
What Do I Need to Know to Use REST?

To use REST in web applications, you need to address the following:

- The HTTP method that you need
- The resource identified by its URI
- The content of the request and parsing of the response accordingly

To know how to use a particular REST service, you need to know what resources you need to access, what methods you can ask for, and what data is going to be exchanged.

The preferred format for the data sent from and to servers is JSON, although the specific format is service dependent.



HTTP Methods

GET

Get a resource

Used to get a resource from the server,
e.g. Get user list, or get a the details of a user

POST

Add a resource

Used to create a new resource in the server,
e.g. add a new user

PUT

Update a resource

Used to update a resource in the server,
e.g. update the user details

DELETE

Delete a resource

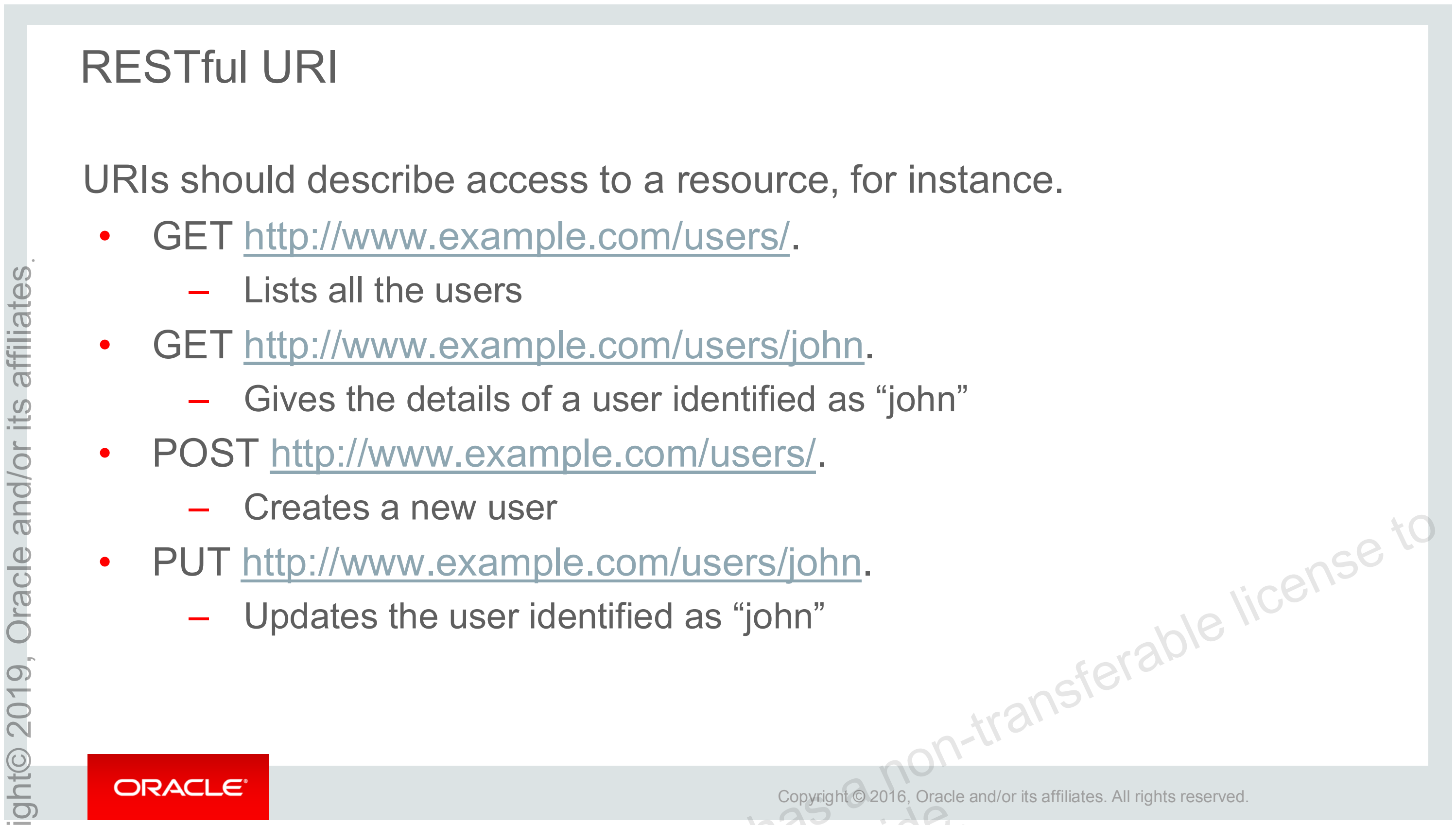
Used to delete a resource in the server,
e.g. delete or disable a user

ORACLE

- `GET` is used to obtain the resource.
- `PUT` is used to update the resource.
- `POST` is used to add a new resource.
- `DELETE` is used to remove a resource.

HTTP defines the `OPTIONS`, `HEAD`, `TRACE`, and `CONNECT` methods as well but their usage in RESTful APIs is not as wide.

Sometimes, you will find the `OPTIONS` method returning a representation of the possible uses of the other methods as well as the URI compositions available. This is not standard and should not be relied upon.



RESTful URI

URIs should describe access to a resource, for instance.

- GET http://www.example.com/users/.
  - Lists all the users
- GET http://www.example.com/users/john.
  - Gives the details of a user identified as “john”
- POST http://www.example.com/users/.
  - Creates a new user
- PUT http://www.example.com/users/john.
  - Updates the user identified as “john”

ORACLE

In the examples in the slide, you can see the common structure of the RESTful API’s URIs.

URIs are usually constructed progressively as you need more details.

For an application that stores places and reviews of such places:

- `/nearby/places`: Refers to the places that are nearby to the user
- `/places`: Refers to the list of places registered
- `/places/toms_pizza`: Refers to the place called Tom’s Pizza
- `/places/toms_pizza/reviews`: Refers to the reviews of the place called Tom’s Pizza

You can see how methods would apply:

- `GET /nearby/places`: Gets a list of the nearby places
- `POST /places`: Adds a new place
- `PUT /places/toms_pizza`: Updates the place called Tom’s Pizza
- `DELETE /places/toms_pizza`: Deletes the place called Tom’s Pizza
- `DELETE /nearby/places`: NOT ALLOWED

Quiz

Which of the following are valid HTTP methods?

a. GET
b. PUSH
c. UPDATE
d. POST
e. DELIVER
f. DELETE
g. All of the above

Quiz

Which of the following are true of REST

a. Uses HTTP
b. Is used by web browsers
c. Provides architecture used by the web
d. Performs stateless operations
e. All of the above

Quiz

What is JSON?

a. An alternative to XML
b. A machine-readable script that is not human-readable
c. A subset of Java Script
d. Defines key-value pairs
e. Used to represent data and functions
f. All of the above

Summary

In this lesson, you should have learned how to:

- Define HTTP and HTML
- Describe the steps in an HTTP transaction
- List key HTTP headers in requests and responses
- List the HTTP Methods
- Define the structure of a URL and a URI
- Describe a REST Web Service
- Describe the characteristics of JSON