Scraping your First Web Page with Python

GETTING STARTED WITH WEB SCRAPING

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Understanding HTTP for accessing web content

Fetching web content using HTTP

Choosing between different Python HTTP libraries

Working with GET, PUT and POST requests

Understanding and handling URL redirects

Prerequisites and Course Outline

Prerequisites

Basic Python programming

Basic knowledge of HTML, CSS and web pages

Prerequisite Courses

Python Fundamentals

Your First Day with HTML

Course Outline

Getting started with web scraping

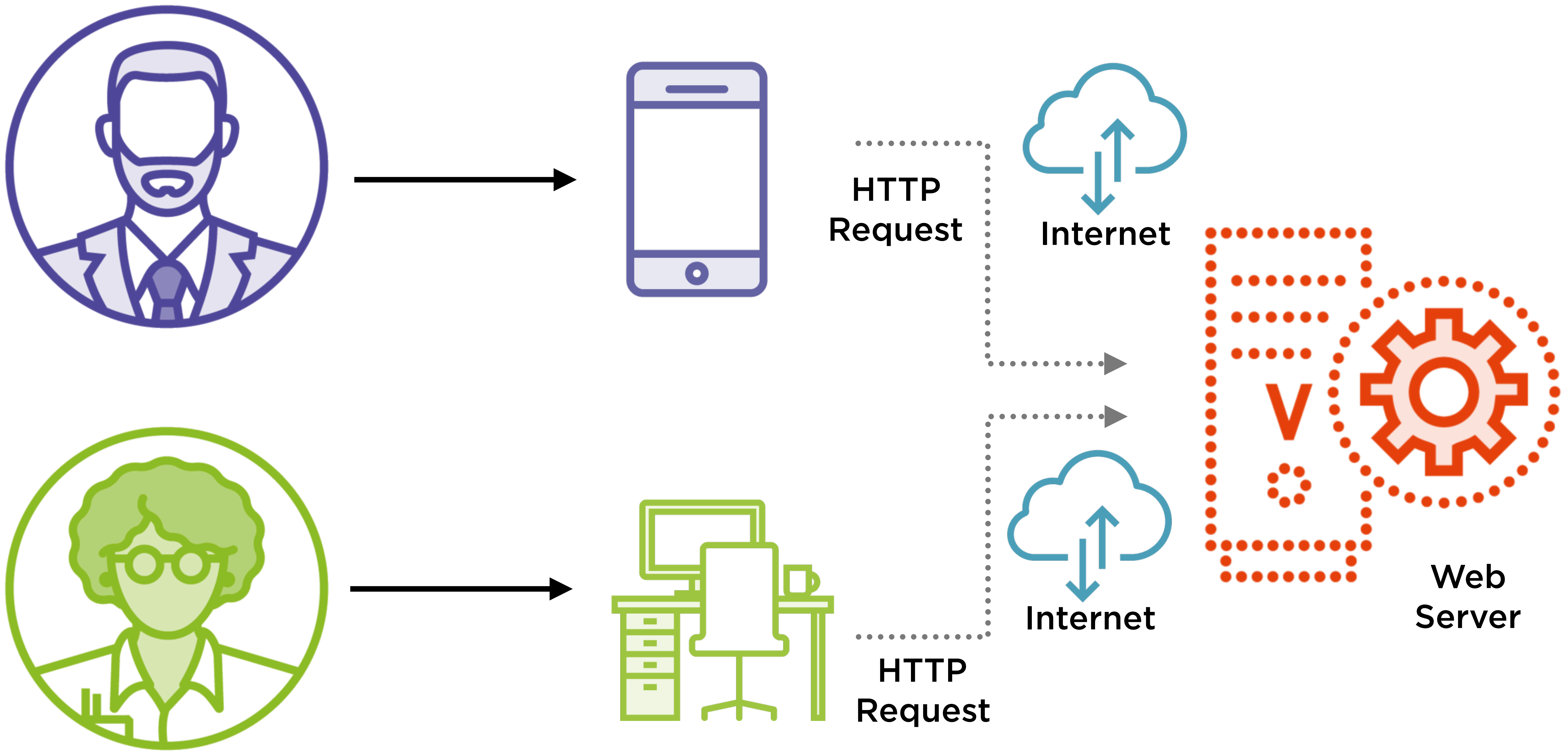
Working with the parse tree in Beautiful Soup

Selecting elements using the Scrapy shell

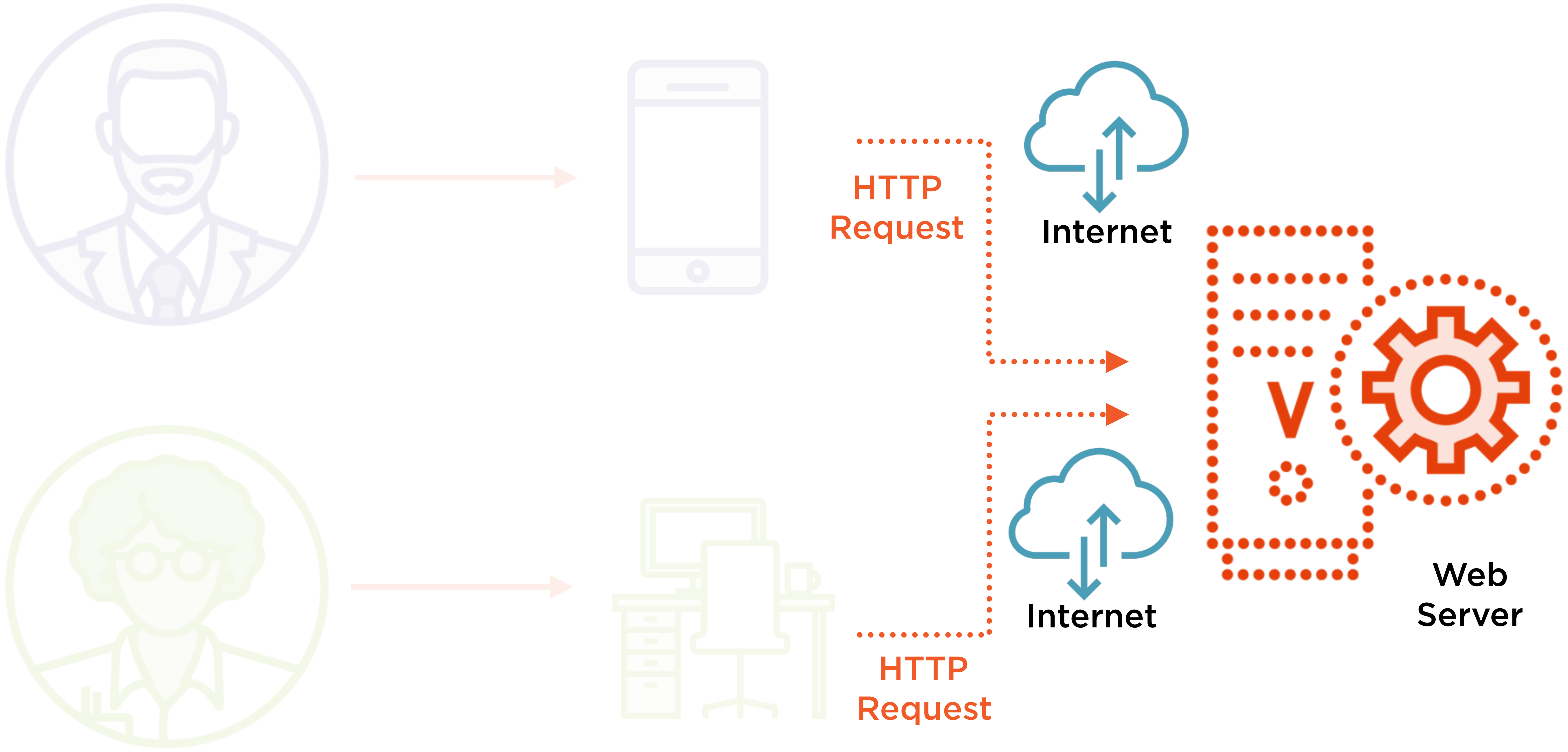
Automated web scraping using Scrapy spiders

Accessing Web Content

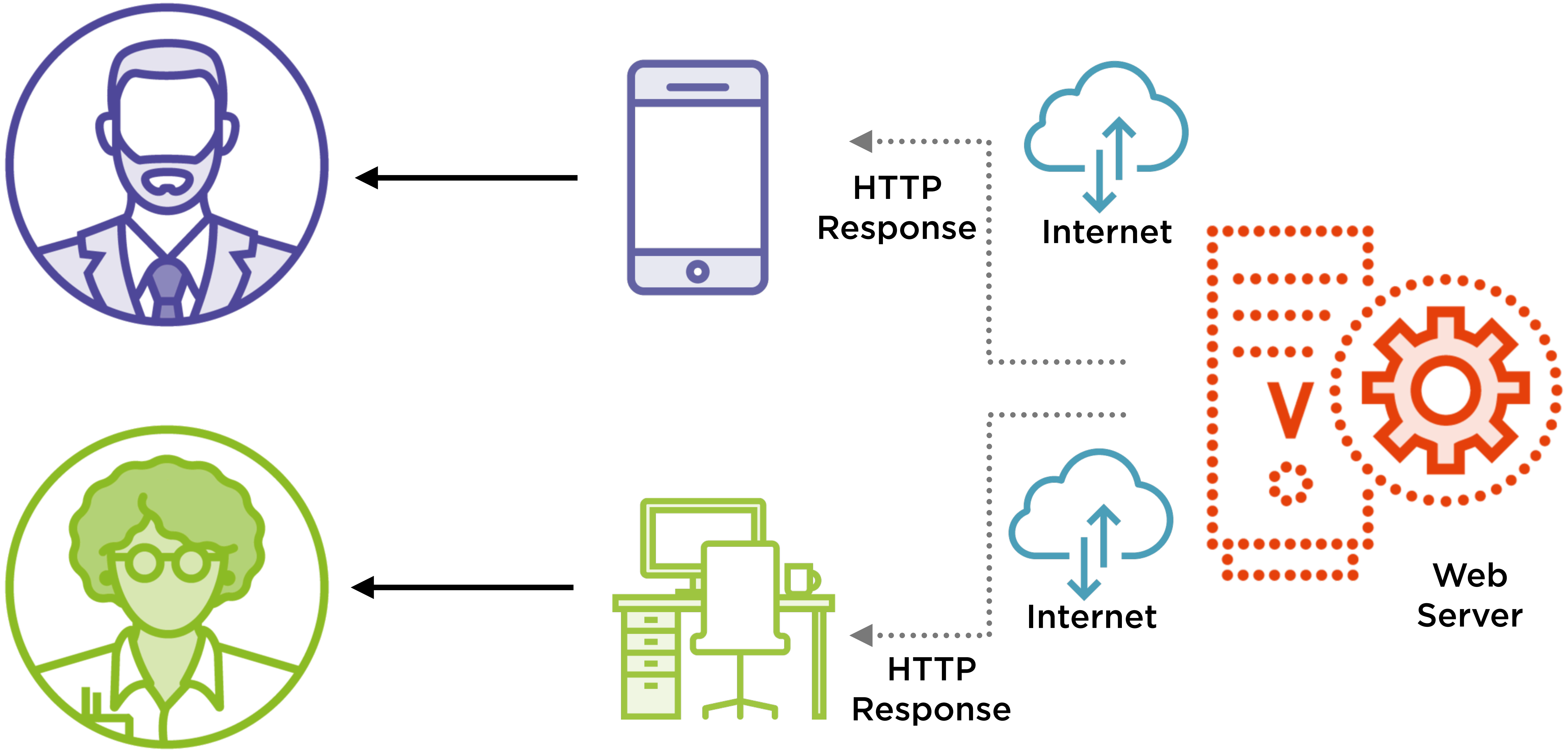
Accessing Web Content



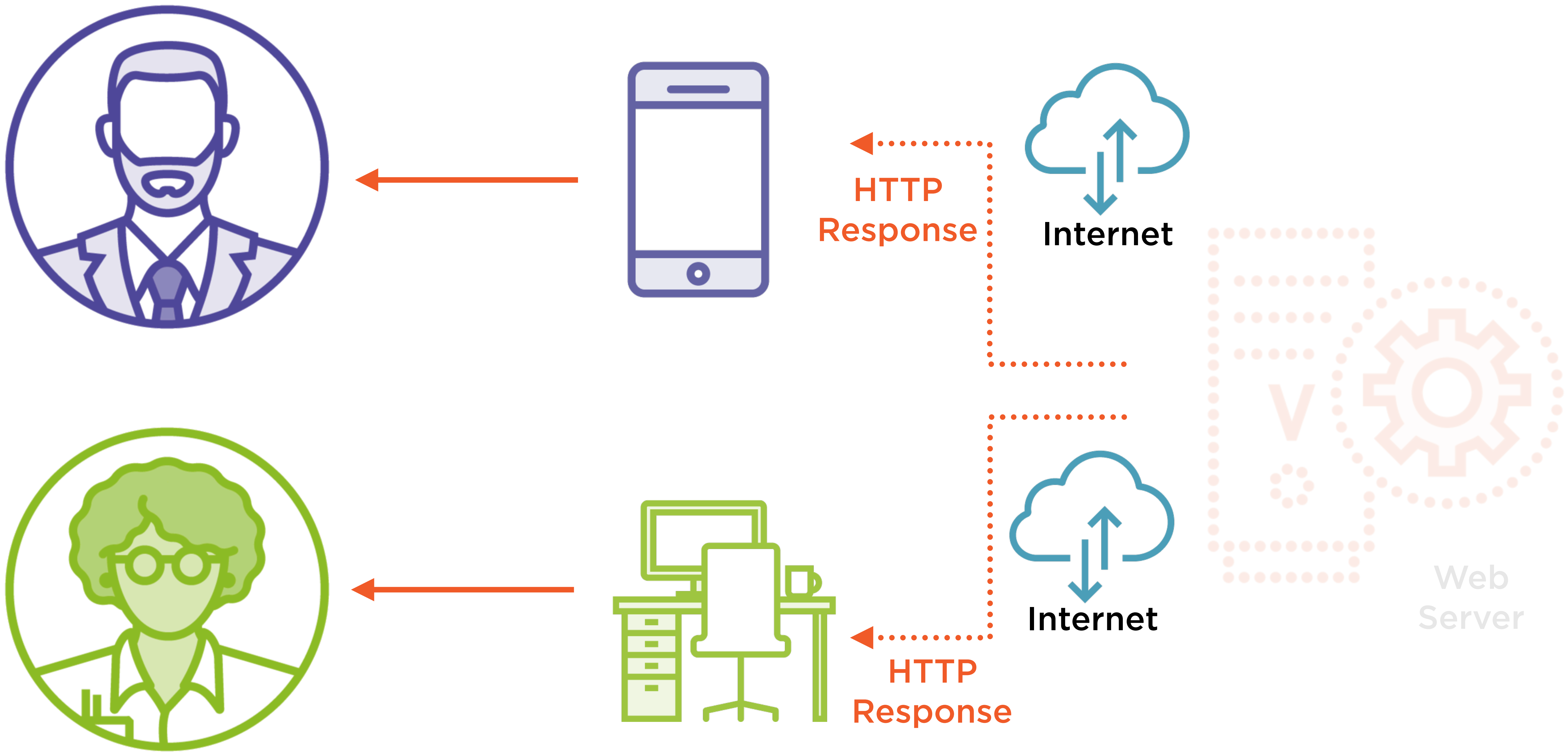
Accessing Web Content



Accessing Web Content



Accessing Web Content



Hypertext Transfer Protocol (HTTP)

Simple, ubiquitous text-based protocol used by browsers and apps to access web content

HTTP Clients

Client-server protocol

Clients

- Web browser
- Mobile apps
- Programs

HTTP Servers

Host web pages and web content

Maybe static or dynamic

Run HTTP servers such as

- Apache
- nginx
- IIS (Microsoft)
- GWS (Google)

HTTP Requests

Clients make HTTP requests

- GET to fetch resources
- POST to create/update resources
- PUT to idempotently create/update
- HEAD to get only HTTP header
- DELETE to delete resources

HTTP Responses

Servers are standing by to field requests

Send back HTTP responses

HTTP response includes

- Status line with code such as 404, 200
- Response header with metadata
- Response body

Scraping Web Content

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Scraping

Automated extraction of data from websites; website content is first fetched (usually using HTTP) and then parsed to extract specific information.

Web Pages

Websites are collections of web pages

Web pages consist of markup e.g. HTML

This markup is understood and rendered by browsers

Fetching and Parsing

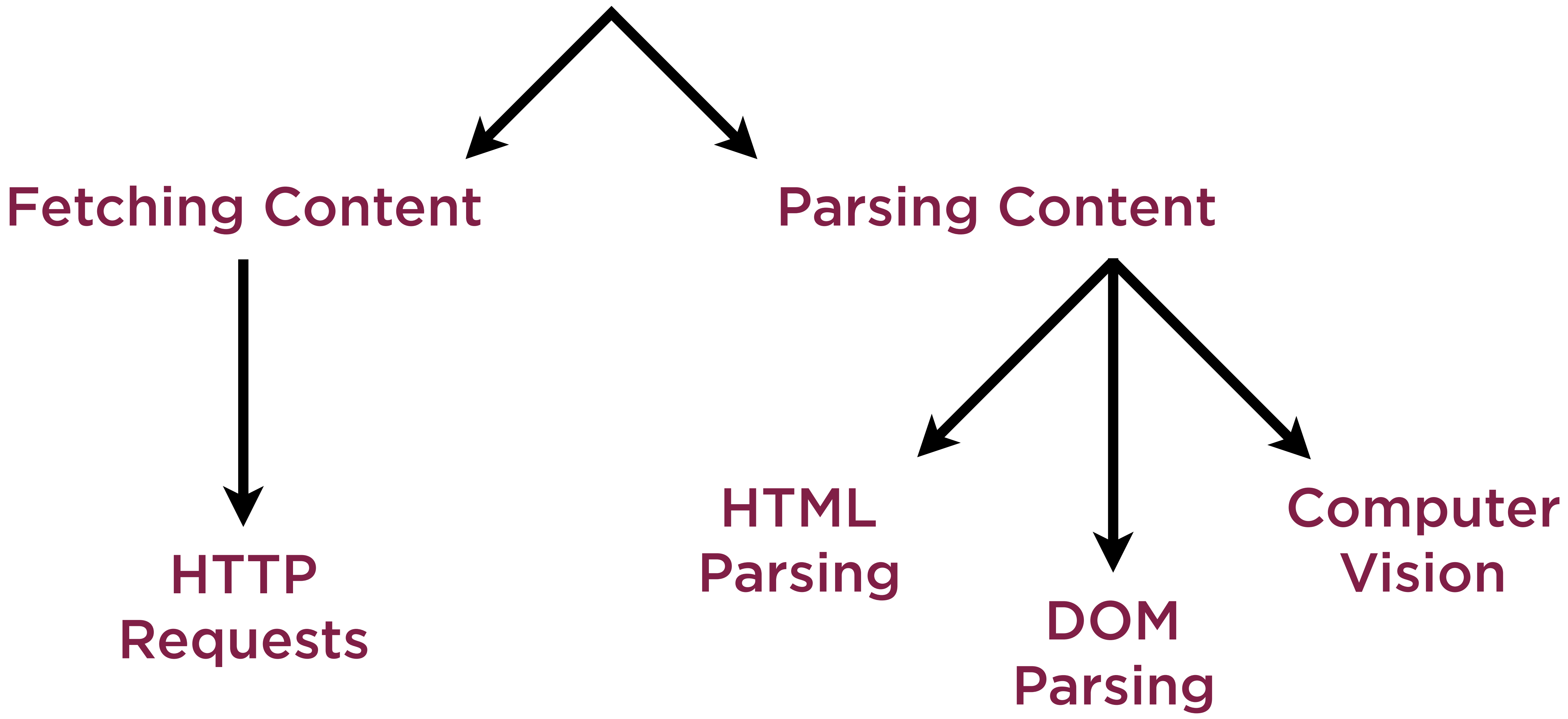
The same HTML markup can be accessed (fetched) via HTTP

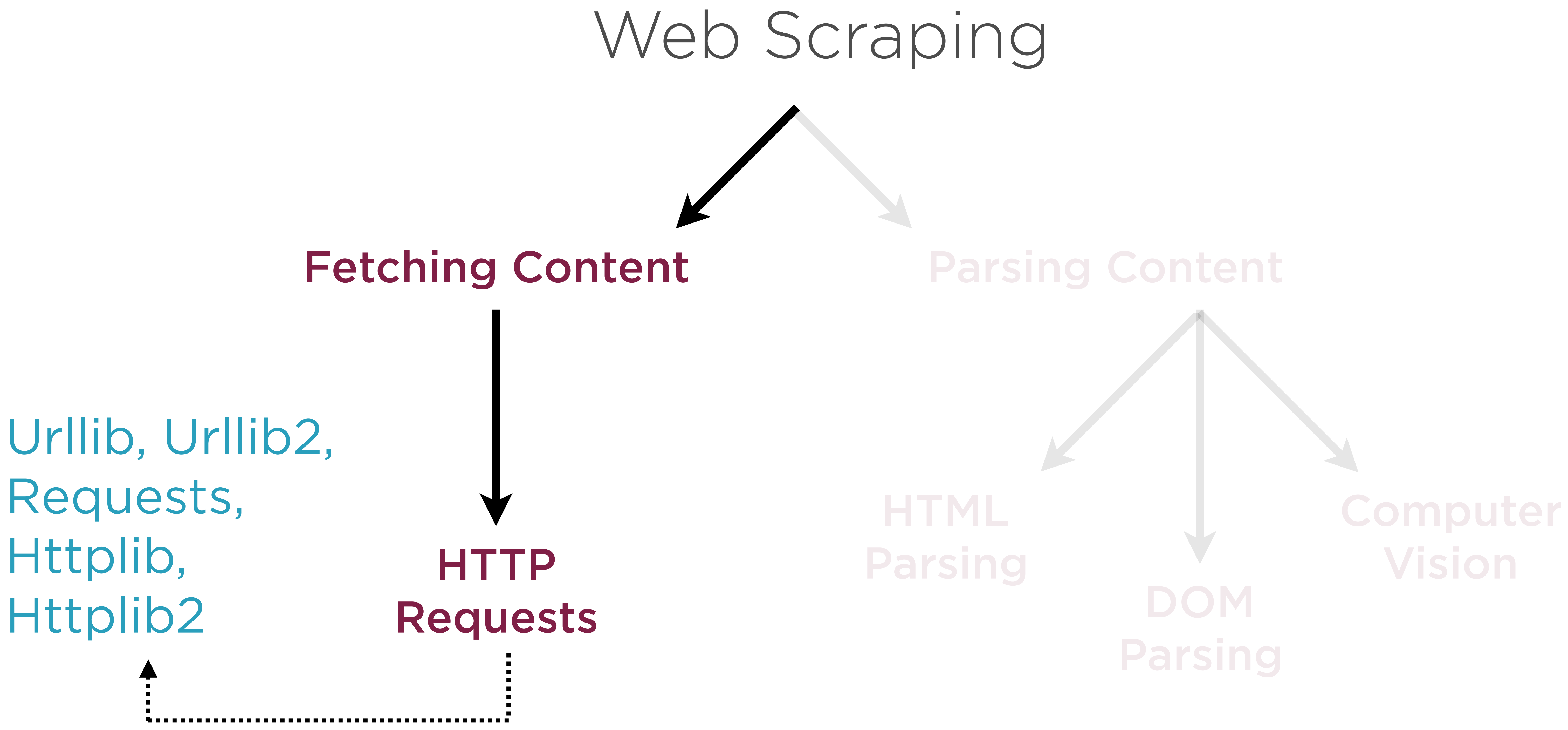
Possesses an in-built hierarchical structure

Parsers can exploit this structure to extract information

Fetching and Parsing Content

Web Scraping




Web Scraping
Fetching Content
Parsing Content
Urllib, Urllib2,
Requests,
Httplib,
Httplib2
HTTP
Requests
HTML
Parsing
DOM
Parsing
Computer
Vision

Python HTTP Libraries

Requests

Httplib2

Httplib

Urllib

Urllib2

HTTP Libraries in Python

Requests

Httplib2

- Httplib

Urllib

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2

- Httplib

Urllib

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Google’s version of Httplib

- Httplib

Urllib

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- Httplib

Urllib

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- Httplib: Little reason to use directly

Urllib

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- ~~Httplib: Little reason to use directly~~

Urllib

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- ~~Httplib: Little reason to use directly~~

Urllib: Part of Python standard library

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- ~~Httplib: Little reason to use directly~~

Urllib: Four distinct namespaces

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- ~~Httplib: Little reason to use directly~~

Urllib: In Python 3.x, subsumes old urllib2

- Urllib2

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- ~~Httplib: Little reason to use directly~~

Urllib: In Python 3.x, subsumes old urllib2

- Urllib2: Python 2 only

HTTP Libraries in Python

Requests: Should be your default choice

Httplib2: Fine-grained control

- ~~Httplib: Little reason to use directly~~

Urllib: In Python 3.x, subsumes old urllib2

- ~~Urllib2: Python 2 only~~

Fetching Web Content

Web servers make content available on HTTP endpoints

Browsers make HTTP requests under-the-hood to get web pages

Web scraping usually involves making such requests programmatically

Many libraries and utilities available

Fetching Web Content

Command-line HTTP requests

- cURL

Python libraries for programmatic access

- Requests
- Httplib2
- Urllib

Demo

Introducing httplib2 and making HTTP GET requests using httplib2

Demo

Making OPTIONS, HEAD, POST and PUT requests with httplib2

Handling redirects with httplib2

Demo

Using the urllib library for HTTP requests

Demo

Using the high-level requests library for GET and POST requests

Demo

Handling redirects using the requests library

Summary

Understanding HTTP for accessing web content

Fetching web content using HTTP

Choosing between different Python HTTP libraries

Working with GET, PUT and POST requests

Understanding and handling URL redirects