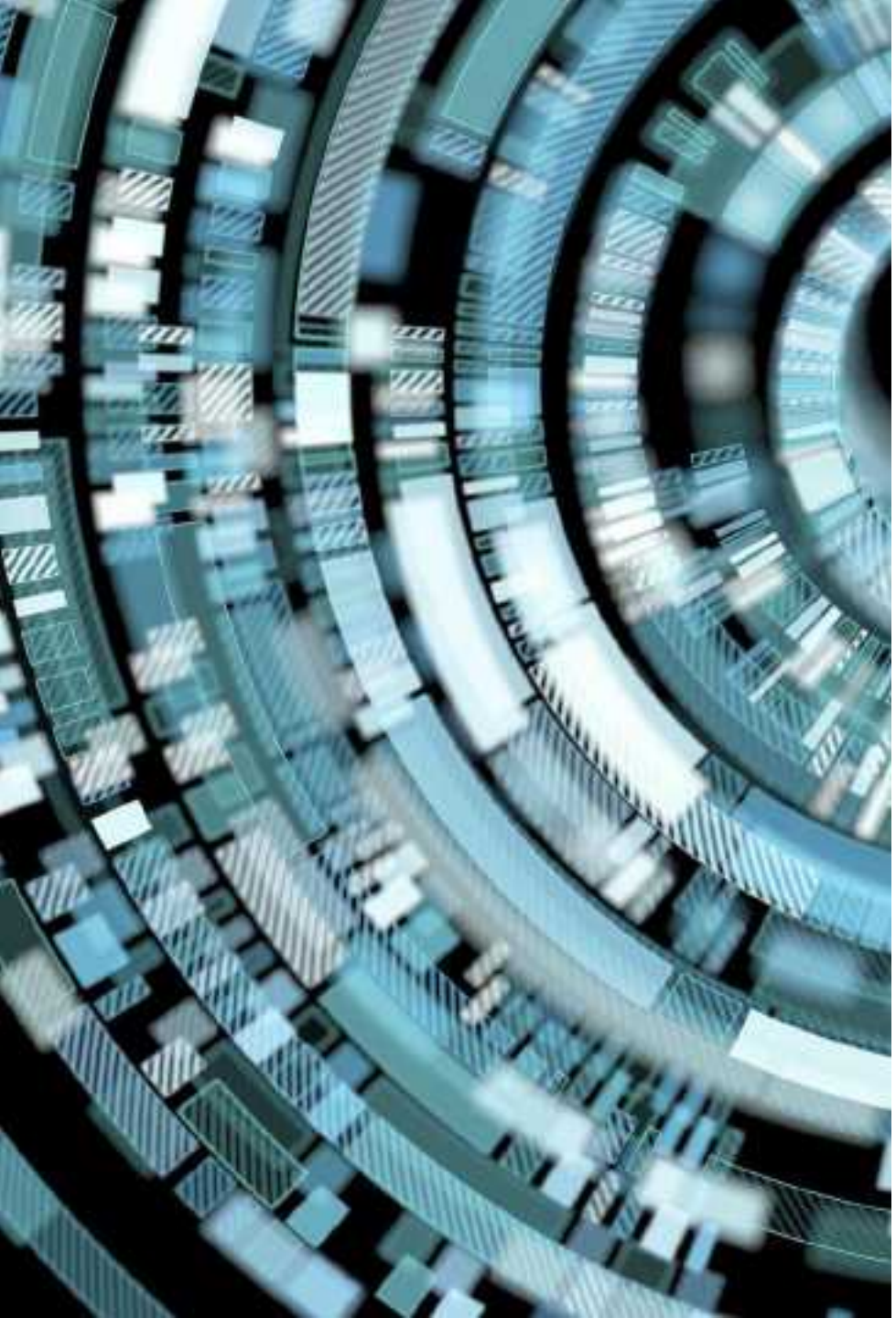
Computer System Design & Application

计算机系统设计与应用A

陶伊达 （TAO Yida）
taoyd@sustech.edu.cn

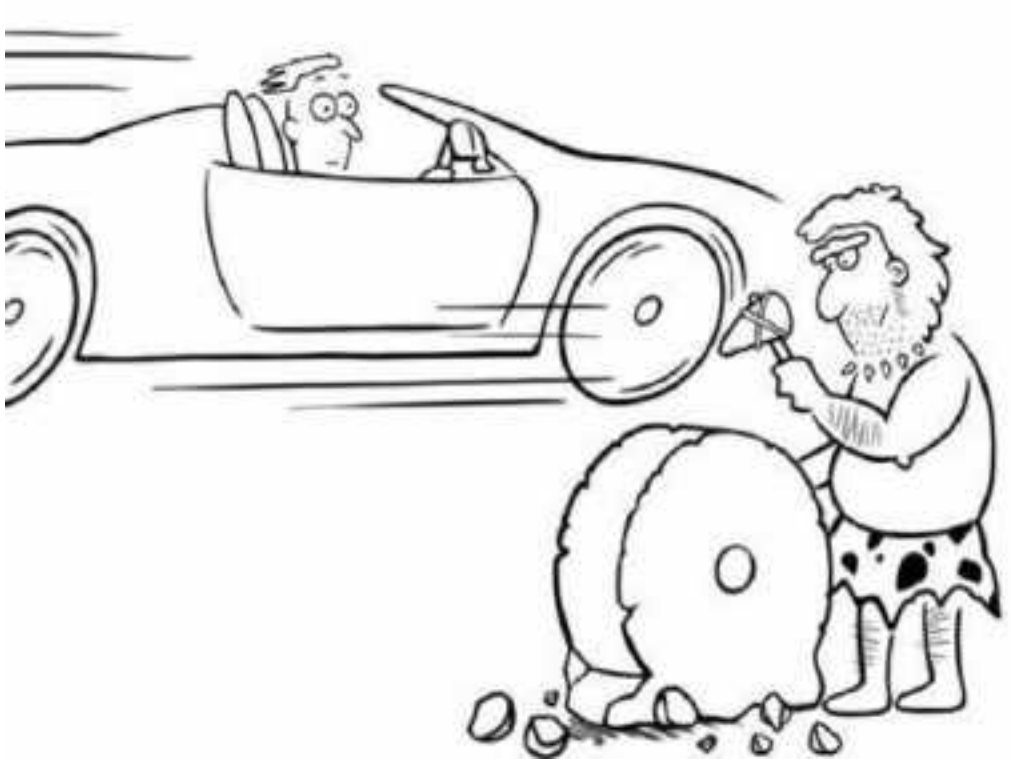


Lecture 7

- Reusable Software
- Web Crawling Libraries
- RESTful API

What is Software Reuse?

- A term used for developing the software by using the existing software components/assets.



https://medium.com/on-technology/reinventing-the-wheel-f4a2152d9f27

What is Software Reuse?

- A term used for developing the software by using the existing software components/assets.

- Reusable Software Assets
  - A cohesive collection of artifacts that solves a specific problem or set of problems encountered in the software development life cycle
  - A reusable asset is created with the intent of reuse.

http://walderson.com/IBM/Practices/ScalingAgile/core.tech.common.extend_supp-ibm/guidances/concepts/reusable_asset_43D27168.html

Classifying Reusable Assets

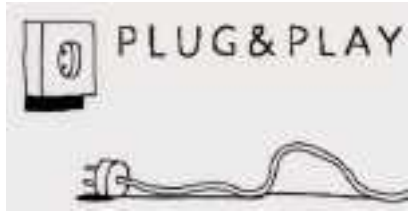
By usage

By level of implementation

By level of abstraction

Reusable Assets by Usage

- Black box reuse
  - Reuse of an asset as is
  - No modification of the asset is needed (plug & play)



- Glass box reuse
  - Modification of asset is needed in order to use it for the specific problem

http://www.cs.kent.edu/~jmaletic/cs63901/lectures/ReusableAssets.pdf

Reusable Assets by Level of Implementation

- **No Implementation**
  - These assets have no implementation, and are represented in an abstract form (e.g., Design Patterns)
- **Partial Implementation**
  - These assets are considered partial implementations, but have a variability point and require additional elements before they can be instantiated (e.g., Frameworks)
  - Variability point: a location in the asset that may have a value provided or customized by the asset consumer.
- **Complete Implementation**
  - These assets are considered to be complete implementations, and can be instantiated as-is, without modification (e.g., Libraries)

http://walderson.com/IBM/Practices/ScalingAgile/core.tech.common.extend_supp-ibm/guidances/concepts/reusable_asset_43D27168.html

Reusable Assets by Level of Abstraction

- Architectures
- Idioms
- Design Patterns
- Frameworks
- Libraries

Architectures

- Software Architecture involves the description of elements from which systems are build, interactions among those elements, patterns that guide their composition, and constraints on these patterns [Shaw96].
- High level of abstraction
- Examples
  - Client-server
  - Pipe-and-filter: a simple architectural style that connects a number of components that process a stream of data, each connected to the next component in the processing pipeline via a Pipe

Idioms

- Typical styles of methods which are used to build a software systems (a philosophy of use)
- High level of abstraction
- Examples
  - Coding styles
  - GUI look and feel

http://www.cs.kent.edu/~jmaletic/cs63901/lectures/ReusableAssets.pdf

Design Patterns

- Description of methods (relations between objects and classes) that can be customized to solve a general, recurring design problem in a particular context.
- High level of abstraction
- Examples
  - Factory method
  - Decorator
  - Strategy

Frameworks

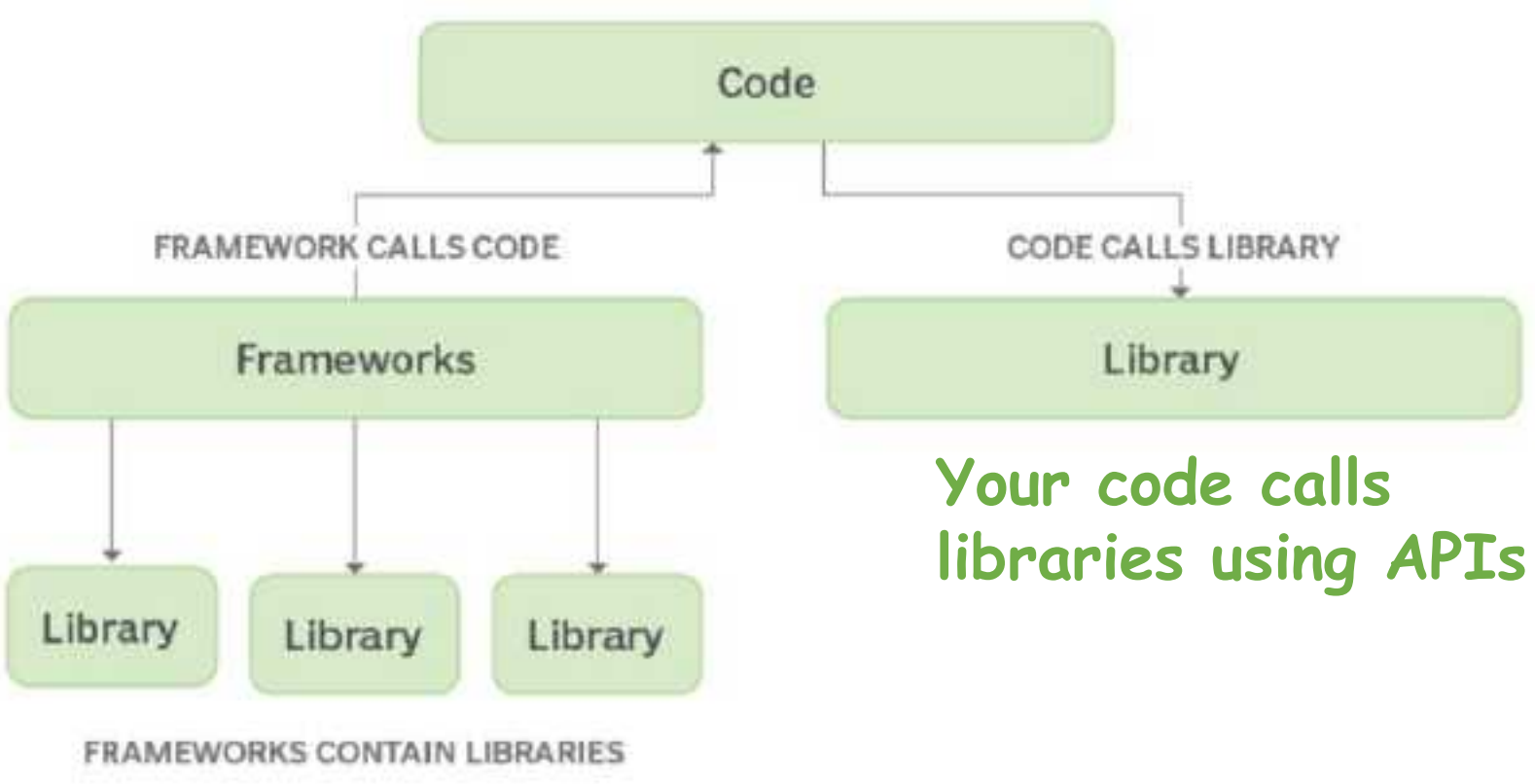
- A set of reusable classes and interfaces which provide a ready-made architecture
- Reusable designs of all/part of system
- To provide a system/application skeleton that developers can customize.
- High/low level of abstraction (are actual programs)
- Examples
  - Java Collections Framework
  - GUI Framework
  - Web Framework

Libraries/Kits

- A set of useful pre-written code, classes, routines, procedures, scripts, configuration data and more
- Low level of abstraction
- Examples
  - Math library
  - Machine learning libraries (e.g., Weka)
  - Logging libraries (e.g., Log4j)
  - Web scraping libraries (e.g., Jaunt, Jsoup)

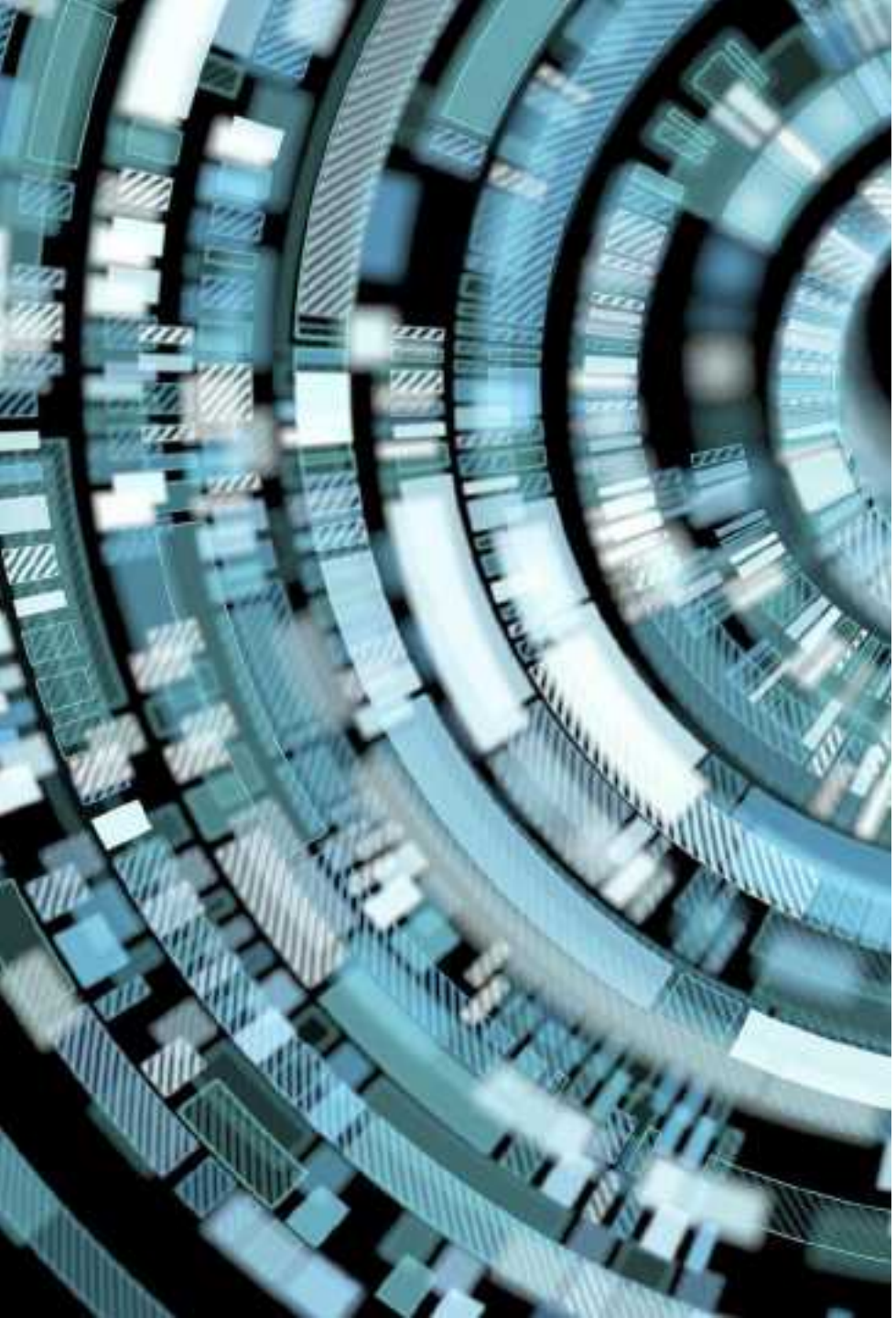
Libraries vs Frameworks vs APIs

"Don't call us, we'll call you."
You have to integrate your code into a framework. Your code has to conform to the framework structure (e.g., implement specific methods or properties).



Your code calls libraries using APIs.

Image: https://www.theserverside.com/tip/Library-vs-framework-How-these-software-artifacts-differ



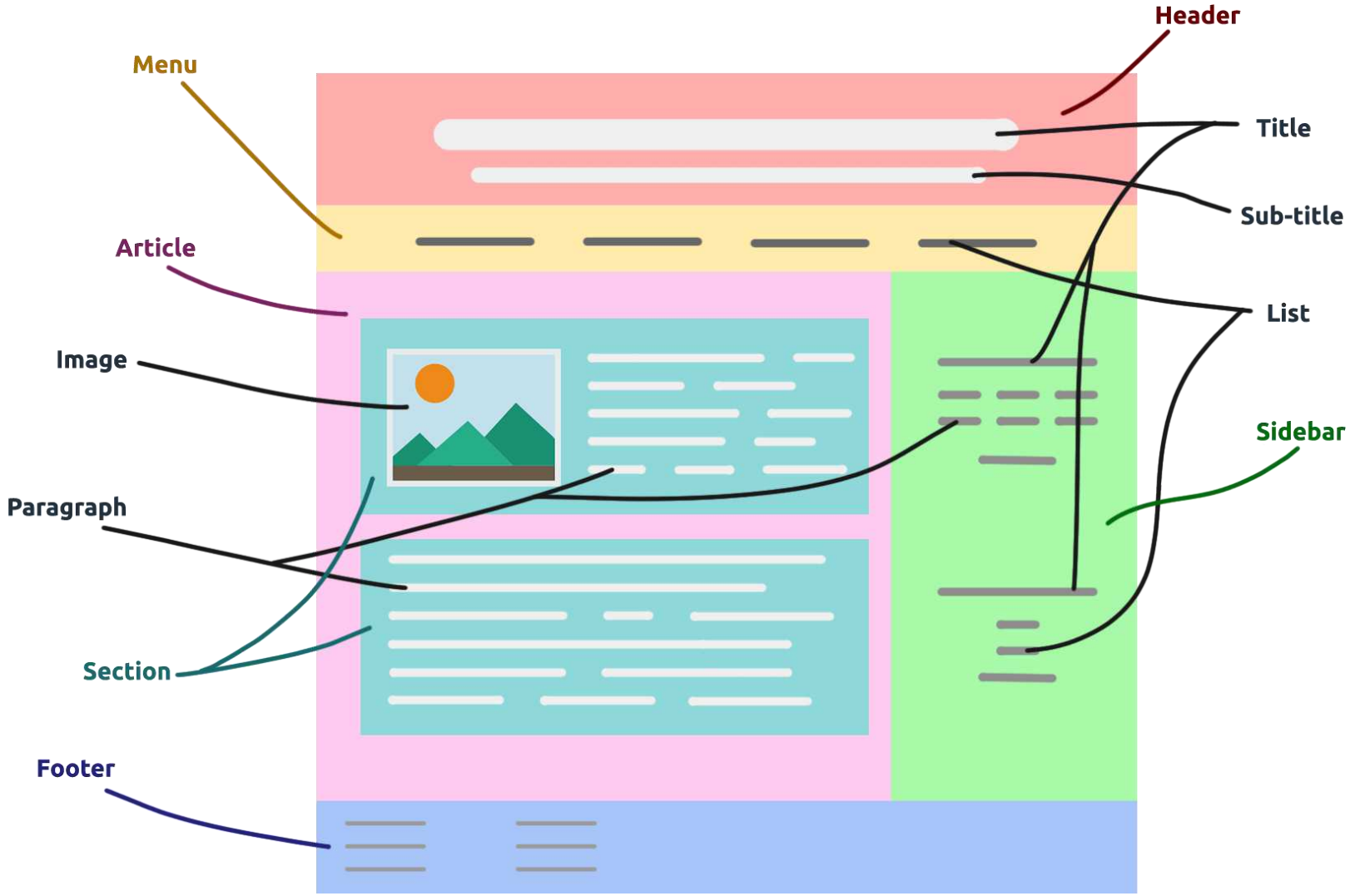
Lecture 7

- Reusable Software
- Case Study: Collecting Website Data
  - Web Scraping Libraries
  - RESTful API

Web Scraping

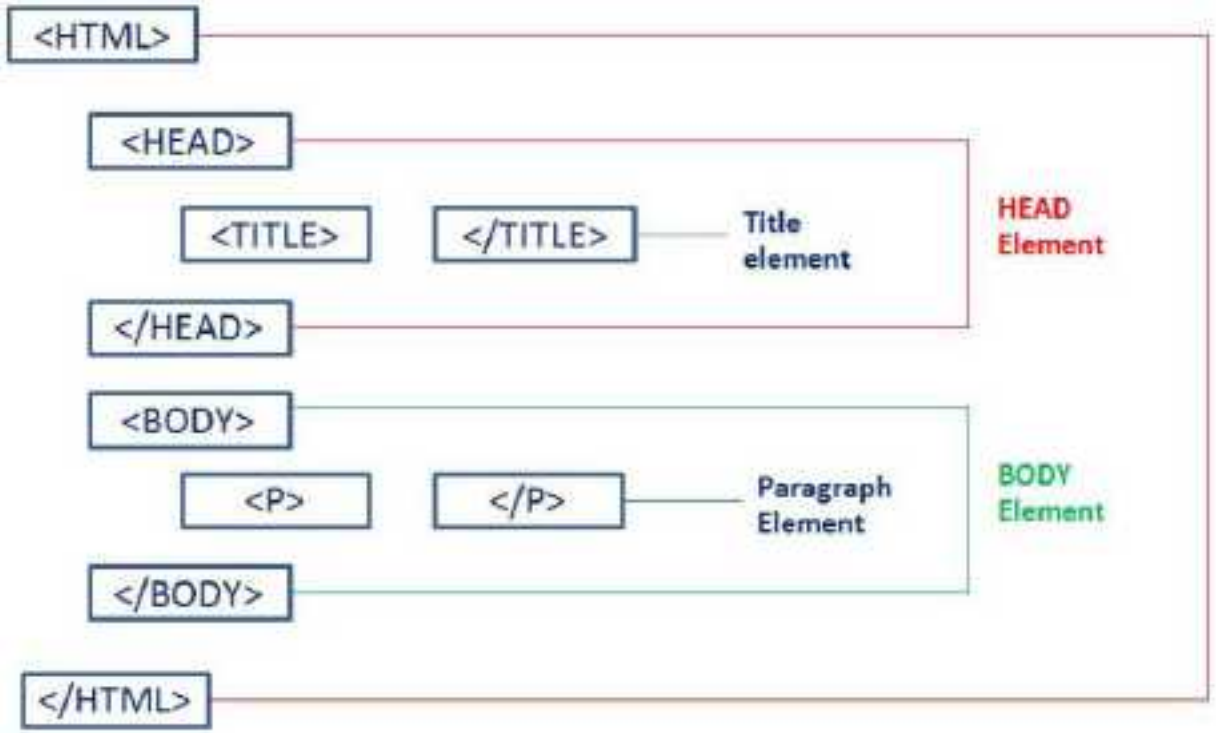
- Web scraping refers to the process of extracting of data from a website or webpage.
- Typically using bots/spiders to navigate through pages and extract data

Web pages Structures



https://www.development-tutorial.com/basic-structure-html-page/

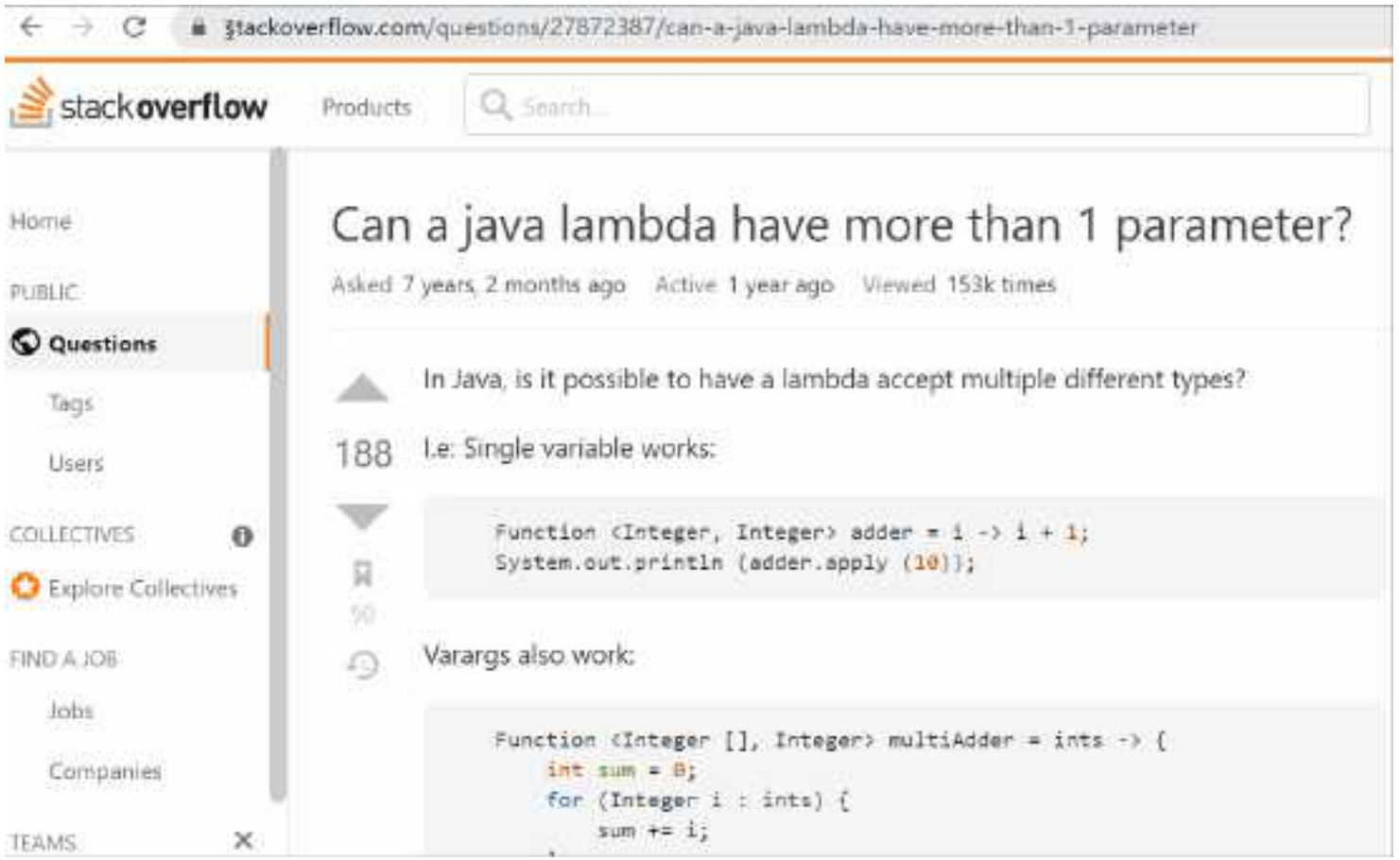
How are web pages created?



https://www.etutorialspoint.com/index.php/basic-html/html-elements

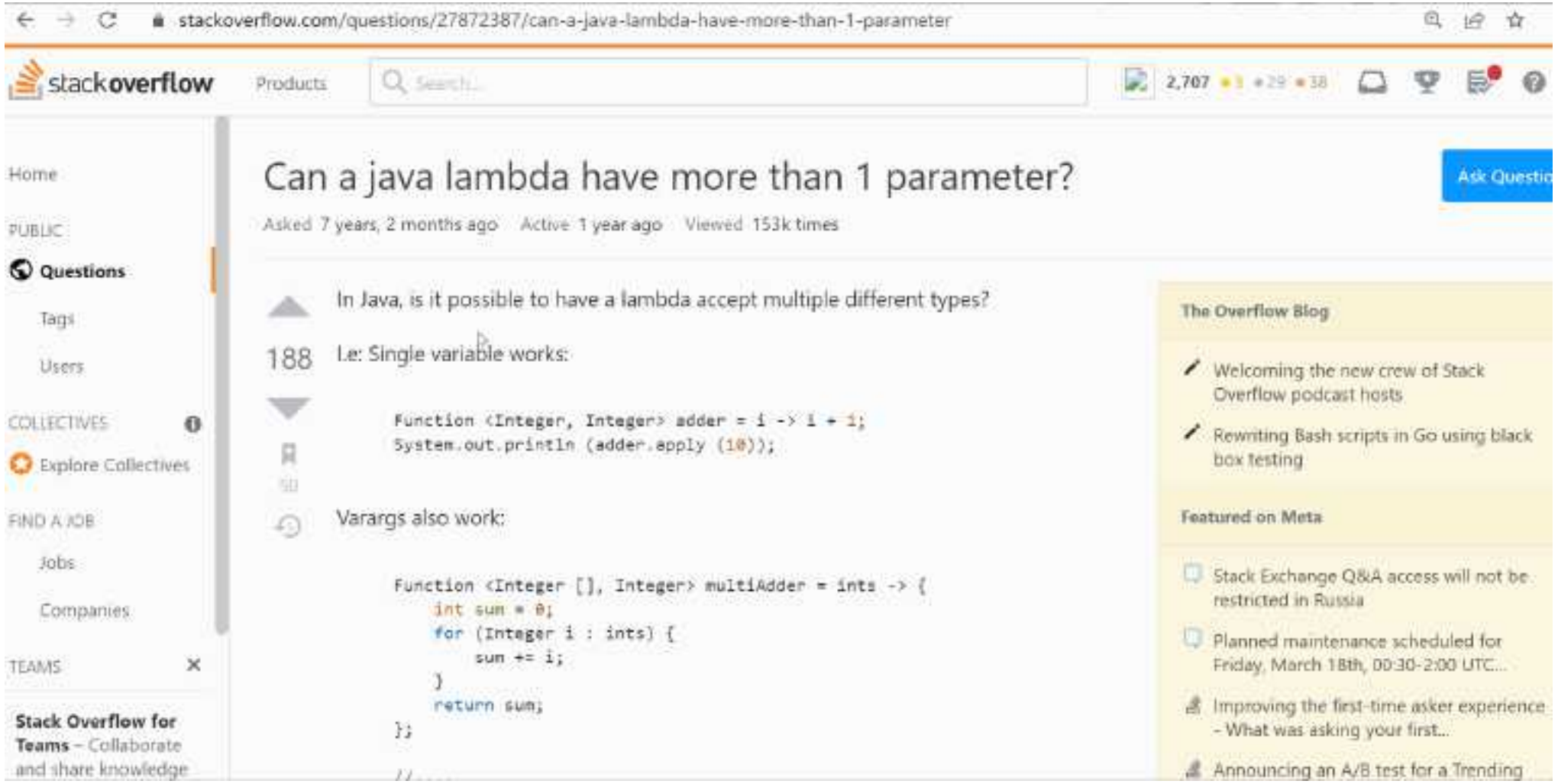
- HTML (Hypertext Markup Language): a hypertext markup language for creating web pages
- HTML uses tags for titles, headings, paragraphs, lists, tables, embedded images, etc., to describe the structure of a web page

Viewing HTML for a Web Page



What if we want to find the html element for a specific part?

Inspecting the HTML for an element



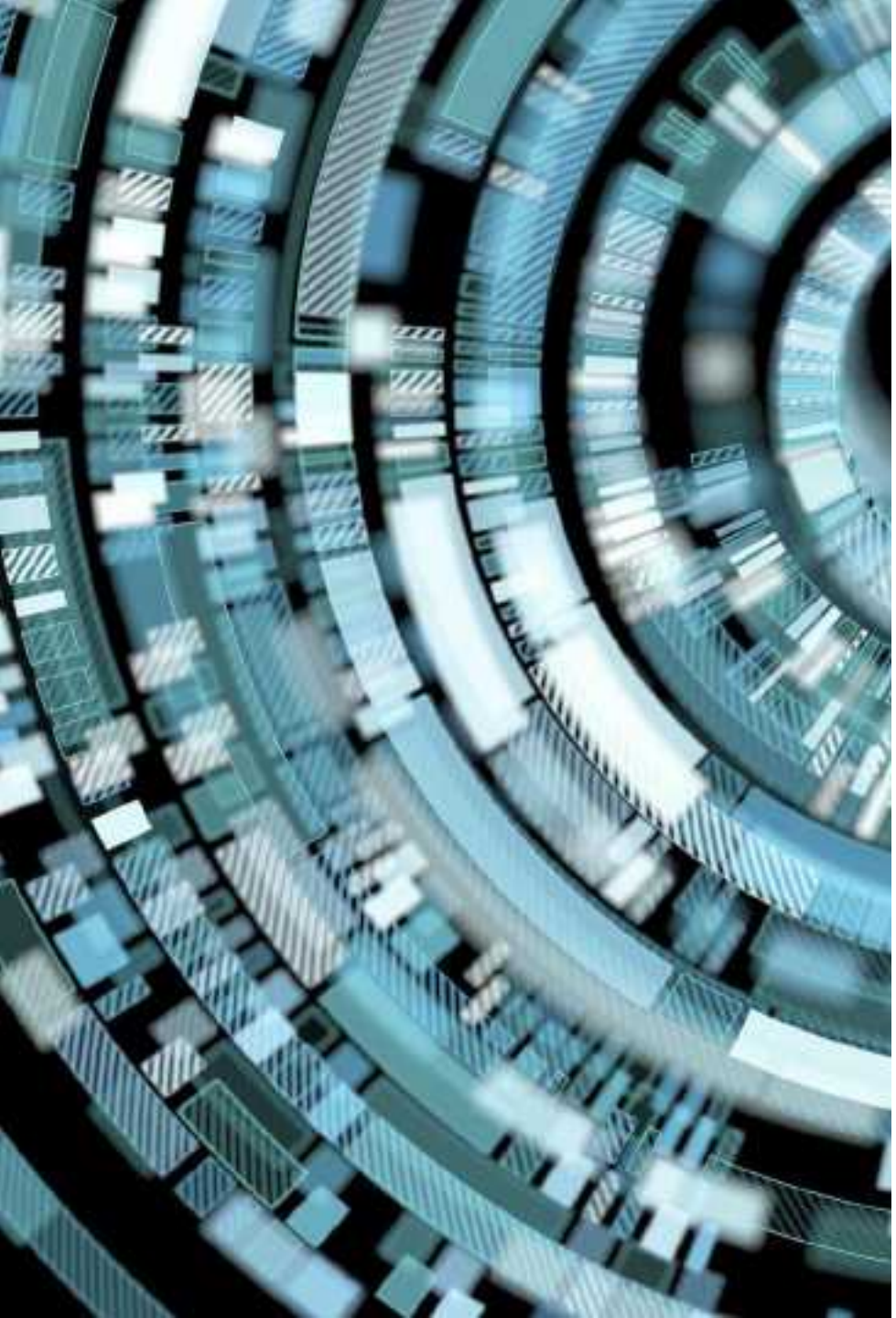
Web-scraping library: Jaunt

- Jaunt is a Java library for web-scraping, web-automation and JSON querying.
- Not the only library for this purpose. Check out others by yourself

```java
import com.jaunt.*;
public class Crawler {
    public static void main(String[] args) throws ResponseException, NotFound {
        UserAgent userAgent = new UserAgent();
        userAgent.visit( url: "https://stackoverflow.com/questions/27872387/");

        // get title
        Element title = userAgent.doc.findFirst("<title>");
        System.out.println(title.getTextContent());

        // get the question's upvote
        Element upvote = userAgent.doc.findFirst("<div itemprop=upvoteCount>");
        System.out.println(upvote.getTextContent());

        // get the upvotes for all the answers
        Elements answers = userAgent.doc
                .findFirst("<div id=answers>")
                .findEvery( query: "<div itemprop=upvoteCount>");
        for(Element answer : answers){
            System.out.println(answer.getTextContent());
        }
    }
}
```

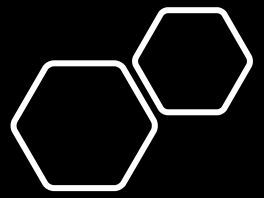
Static vs Dynamic Web Pages

- **Static web pages**
  - Server-side rendered HTML: web page is delivered to the user exactly as stored in the server
  - HTML is fixed
- **Dynamic web pages**
  - JavaScript rendered HTML: web page content is created dynamically using JS
  - HTML is changing (e.g., scrolling down a web page to get the news feed)
  - Needs other advanced scraping strategy/libraries



Lecture 7

- Reusable Software
- Case Study: Collecting Website Data
  - Web Scraping Libraries
  - RESTful API (REST API)



What is REST API?

- **REST**
  - **RE**presentational **S**tate **T**ransfer
  - REST is a software architectural style
- **REST API**
  - A REST API is an API conforms to the constraints of REST architectural style

What are the constraints of REST style?

REST Constraints

- Client-server: A client-server architecture made up of clients, servers, and resources (info like text, image, video)
- Resources could be accessed using URL
- Stateless: Resource requests should be made independently of one another
- Requests are made using HTTP protocol
  - GET: get resources
  - POST: create resources
  - PUT/PATCH: update resources
  - DELETE: delete resources

REST API Request Design

Request = Verb + Object

GET
PUT
PATCH
POST
DELETE

- Typically use noun in plural form, e.g., `questions`
- Exception: `search`
- Allow parameters for filtering, e.g., `?limit=10`

GitHub REST API

URL: https://api.github.com/
Documentation: https://docs.github.com/en/rest

`GET` /repos/{owner}/{repo}

Get a repository info by its owner and repo name

`GET` /repos/{owner}/{repo}/contributors

List repository contributors

`POST` /repos/{owner}/{repo}/issues

Create an issue (must have pull access to this repo)

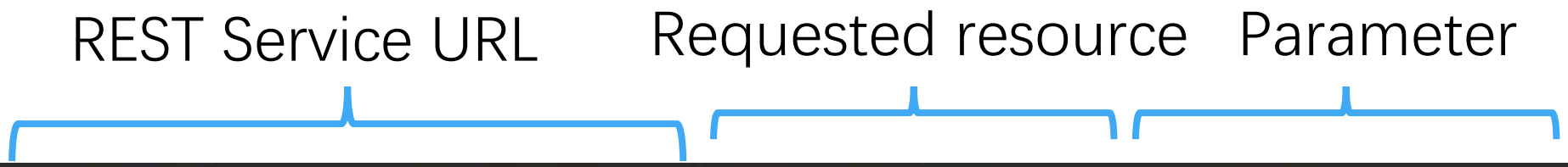
`PATCH` /repos/{owner}/{repo}/releases/{release_id}

Update a release (must have push access to this repo)

`GET` /search/topics

Search for topics (should specify the topic using parameters)

Stack Overflow REST API




```
String s= "https://api.stackexchange.com/questions/27872387?site=stackoverflow";
URL url = new URL(s);
HttpURLConnection conn = (HttpURLConnection)url.openConnection();
conn.setRequestMethod("GET");
conn.connect();

int responseCode = conn.getResponseCode();
String responseMessage = conn.getResponseMessage();
String contentEncoding = conn.getContentEncoding();
```

java.net package

Request verb

200
OK
gzip

Request Response

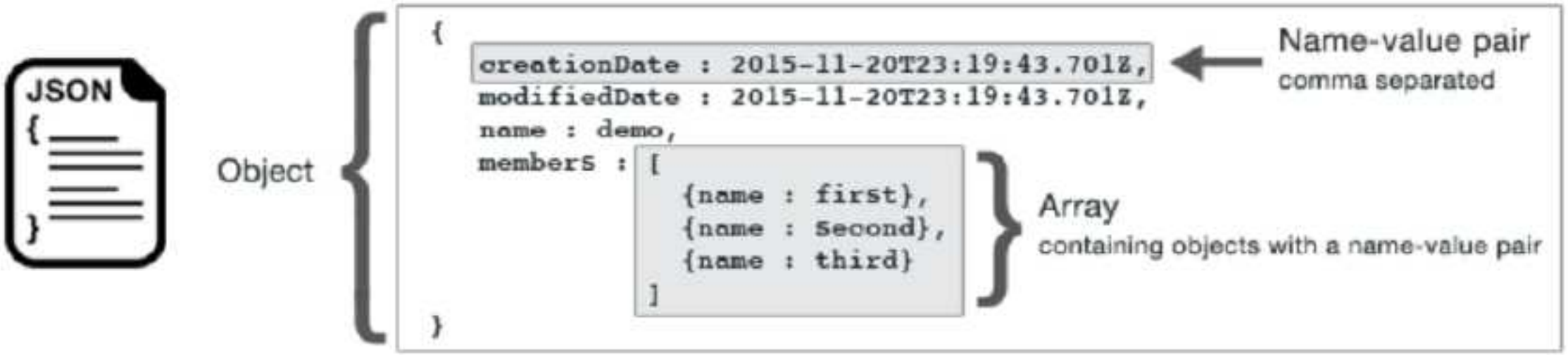
HTTP Status Code

GET /repos/{owner}/{repo}

```
Status: 200 OK
```


```
{
  "id": 1296269,
  "node_id": "MDEwOlJlcG9zaXRvcnkxMjk2MjY5",
  "name": "Hello-World",
  "full_name": "octocat/Hello-World",
  "owner": {
    "login": "octocat",
    "id": 1,
    "node_id": "MDQ6VXNlcjE=",
    "avatar_url": "https://github.com/images/error/octocat_happy.gif",
    "gravatar_id": "",
    "url": "https://api.github.com/users/octocat",
    "html_url": "https://github.com/octocat",
    "followers_url": "https://api.github.com/users/octocat/followers",
    "following_url": "https://api.github.com/users/octocat/following{/other_user}",
    "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}",
```


JSON format

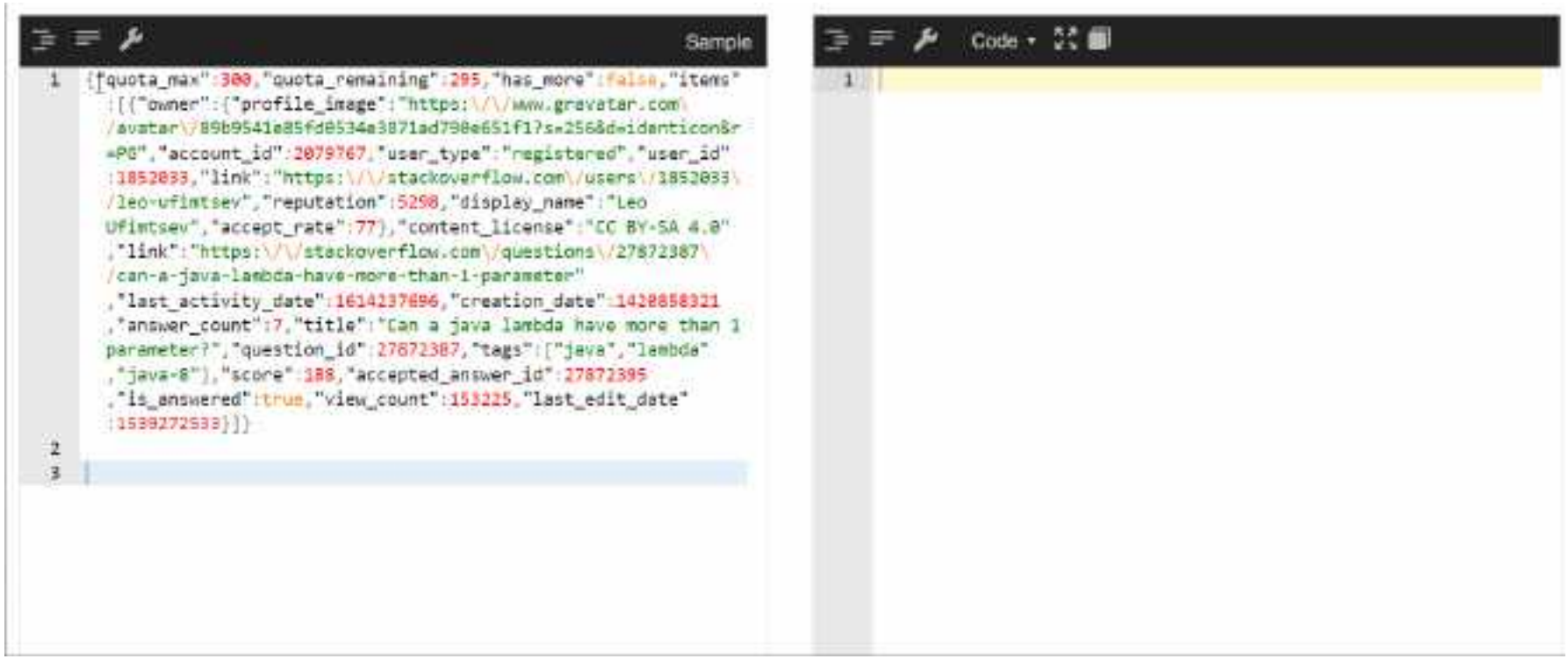


JSON

- JavaScript Object Notation
- An open data interchange format that is both human and machine-readable
- Independent of any programming language

JSON Helper Tools

- Java Libraries (e.g., JSON-simple, GSON, Jackson, etc.)
- JSON viewers (help formatting the JSON string)



REST API IN ACTION

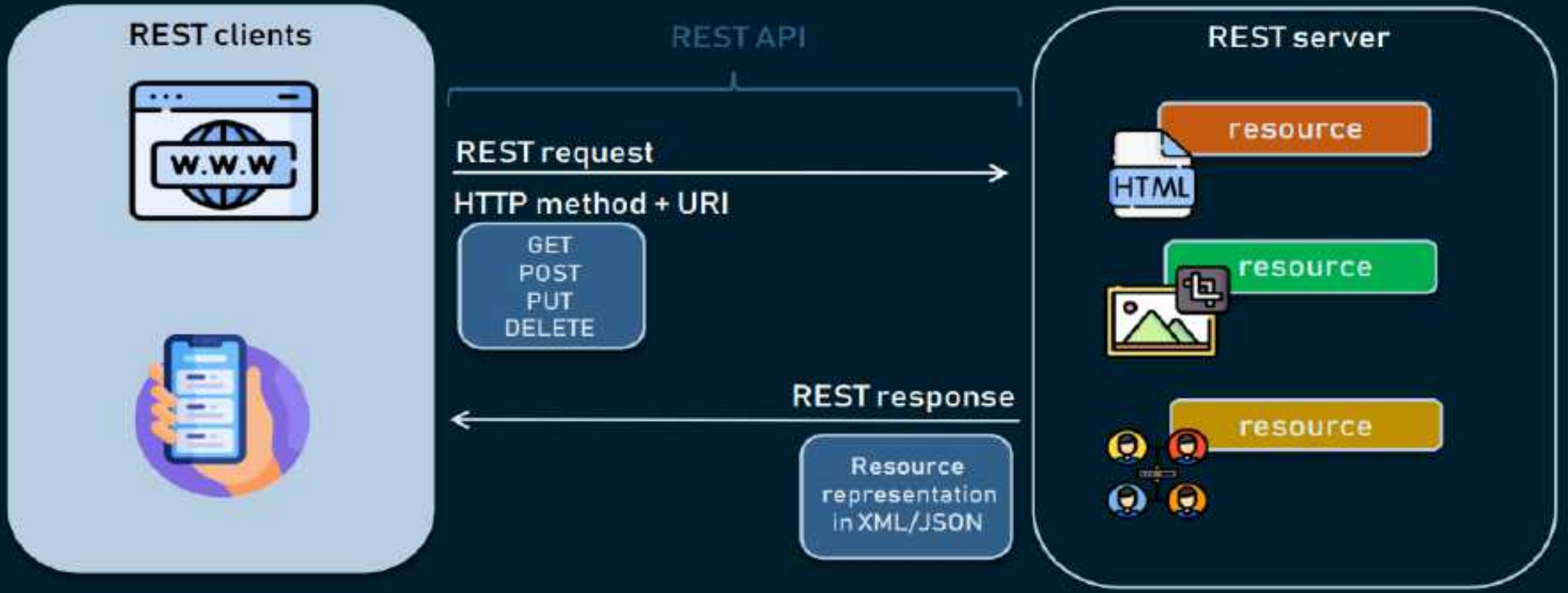


Image source: https://www.altexsoft.com/blog/rest-api-design/

Next Lecture

- Concurrency
- Multithreading