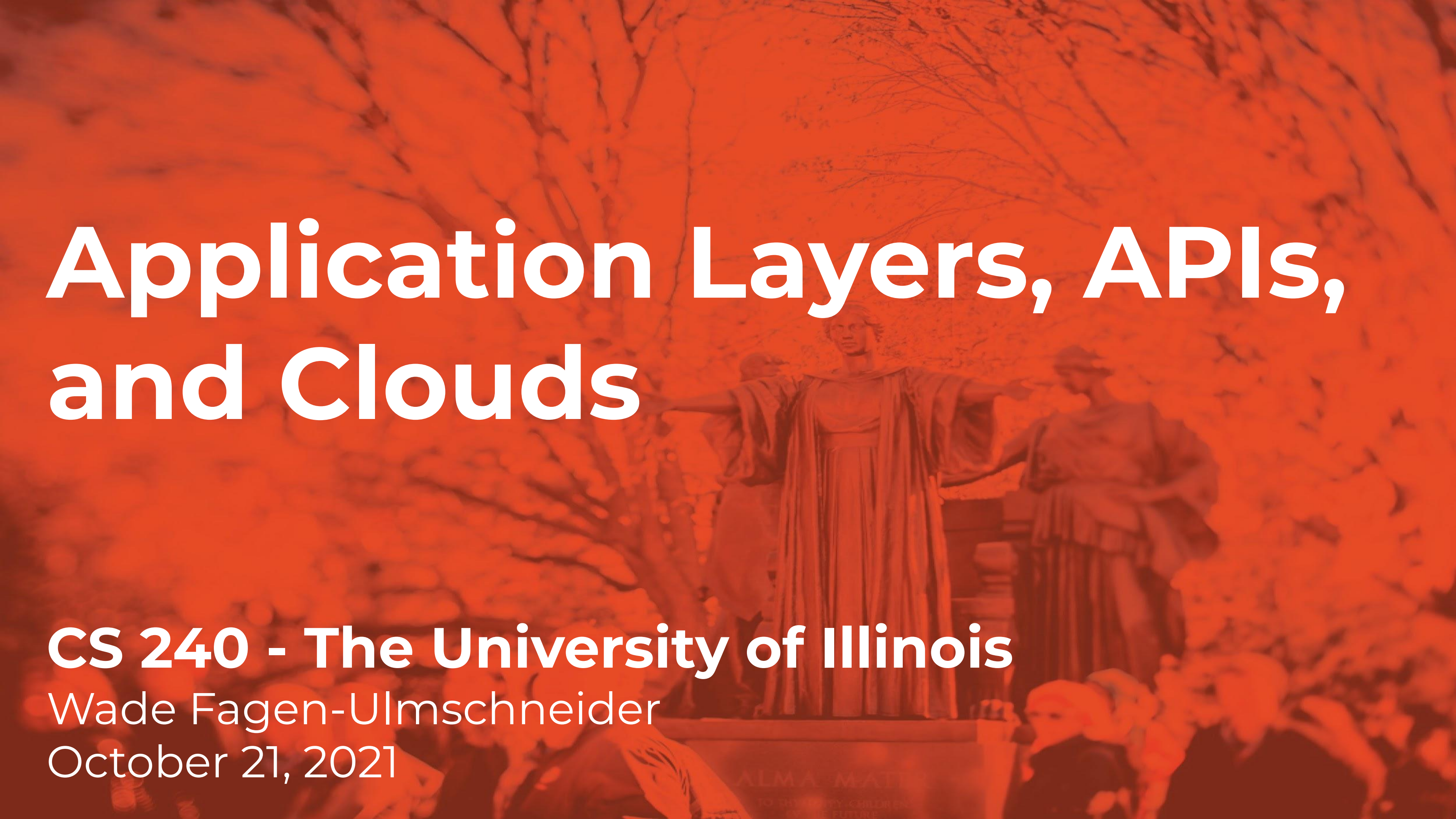
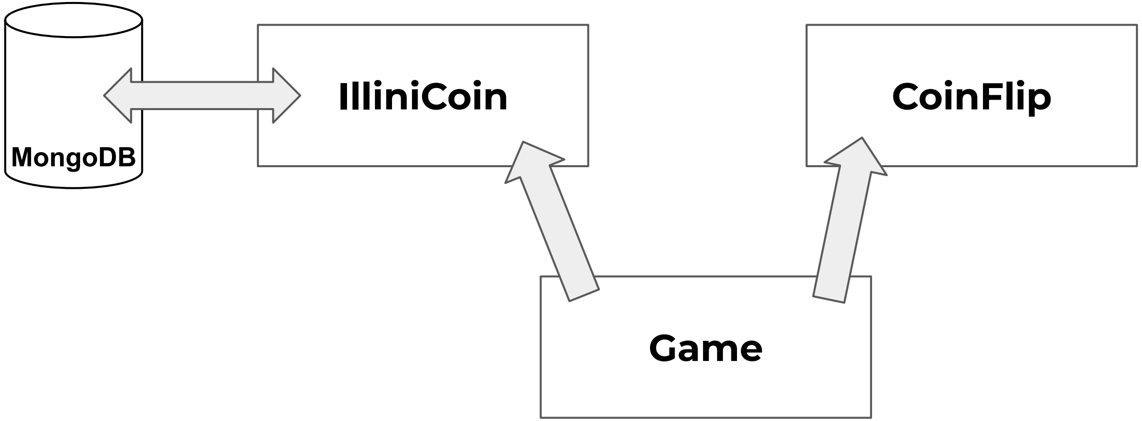


Application Layers, APIs, and Clouds

CS 240 - The University of Illinois

Wade Fagen-Ulmschneider
October 21, 2021


MongoDB
IlliniCoin
CoinFlip
Game

Application Layers

Front-End Software

Front-End Software

What defines "front-end"?

Front-End Software

What are “front-end” bottlenecks?

Front-End Software

What technologies are used for “front-end” development?

Front-End Software

Is there “one” front-end?

Middleware Software

Middleware Software

What defines “middleware”?

Middleware Software

What defines “middleware”?

Middleware Software

What are "middleware" bottlenecks?

Middleware Software

What technologies are used for “middleware” development?

Middleware Software

Is there one middleware?

Middleware Software

How does the middleware interact with the frontend?

Backend Software

Middleware Software

What defines “backend”?

Middleware Software

What are “backend” bottlenecks?

Middleware Software

What technologies are used for “backend” development?

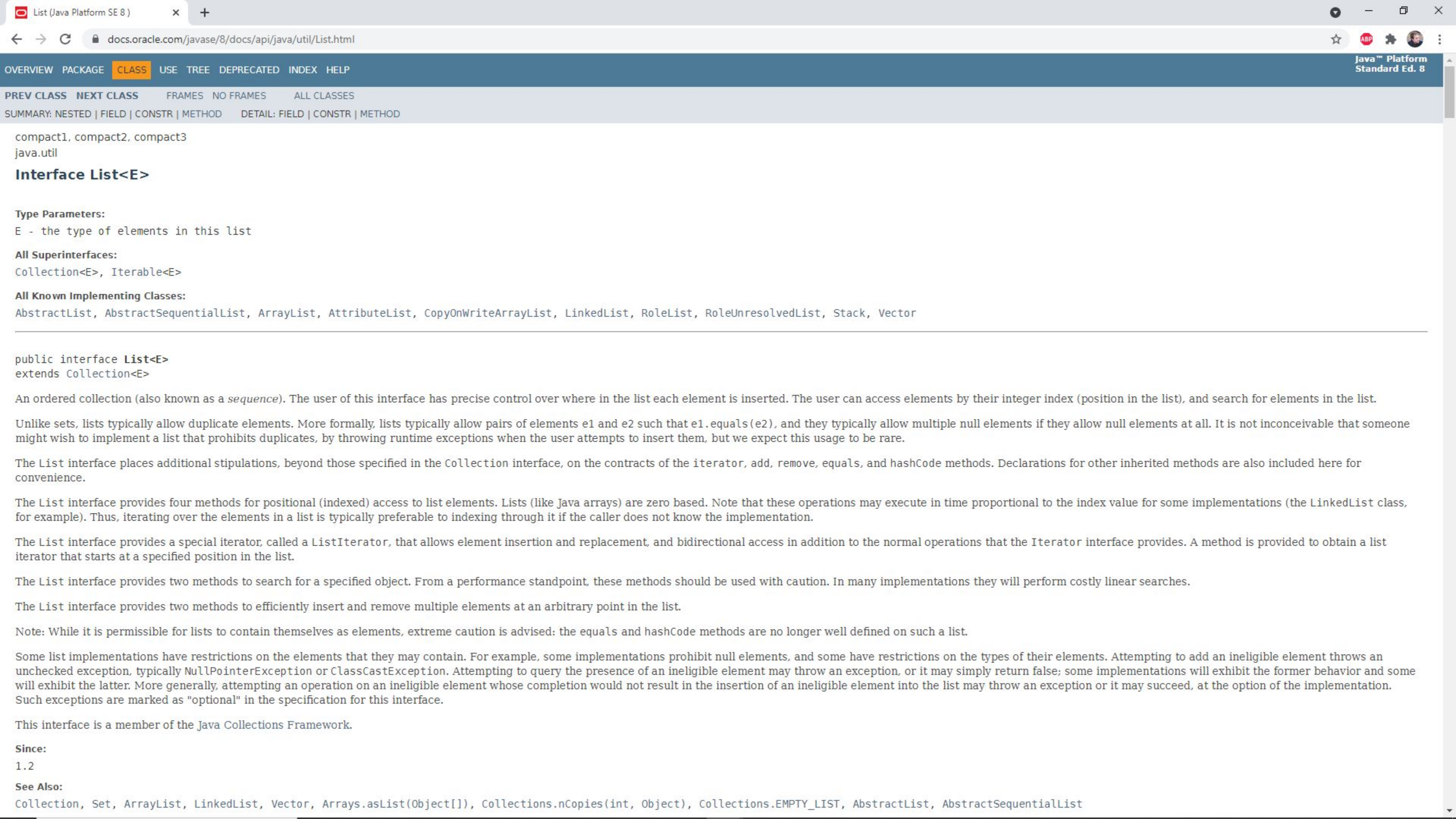
Middleware Software

Is there one backend?

Middleware Software

How does the backend interact with the frontend?

APIs


List (Java Platform SE 8)
docs.oracle.com/javase/8/docs/api/java/util/List.html
OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
Java™ Platform
Standard Ed. 8
PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.util

Interface List<E>

Type Parameters:

`E - the type of elements in this list`

All Superinterfaces:

`Collection<E>, Iterable<E>`

All Known Implementing Classes:

`AbstractList, AbstractSequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector`

```
public interface List<E>
extends Collection<E>
```

An ordered collection (also known as a *sequence*). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements `e1` and `e2` such that `e1.equals(e2)`, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

The `List` interface places additional stipulations, beyond those specified in the `Collection` interface, on the contracts of the `iterator`, `add`, `remove`, `equals`, and `hashCode` methods. Declarations for other inherited methods are also included here for convenience.

The `List` interface provides four methods for positional (indexed) access to list elements. Lists (like Java arrays) are zero based. Note that these operations may execute in time proportional to the index value for some implementations (the `LinkedList` class, for example). Thus, iterating over the elements in a list is typically preferable to indexing through it if the caller does not know the implementation.

The `List` interface provides a special iterator, called a `ListIterator`, that allows element insertion and replacement, and bidirectional access in addition to the normal operations that the `Iterator` interface provides. A method is provided to obtain a list iterator that starts at a specified position in the list.

The `List` interface provides two methods to search for a specified object. From a performance standpoint, these methods should be used with caution. In many implementations they will perform costly linear searches.

The `List` interface provides two methods to efficiently insert and remove multiple elements at an arbitrary point in the list.

Note: While it is permissible for lists to contain themselves as elements, extreme caution is advised: the `equals` and `hashCode` methods are no longer well defined on such a list.

Some list implementations have restrictions on the elements that they may contain. For example, some implementations prohibit null elements, and some have restrictions on the types of their elements. Attempting to add an ineligible element throws an unchecked exception, typically `NullPointerException` or `ClassCastException`. Attempting to query the presence of an ineligible element may throw an exception, or it may simply return false; some implementations will exhibit the former behavior and some will exhibit the latter. More generally, attempting an operation on an ineligible element whose completion would not result in the insertion of an ineligible element into the list may throw an exception or it may succeed, at the option of the implementation. Such exceptions are marked as "optional" in the specification for this interface.

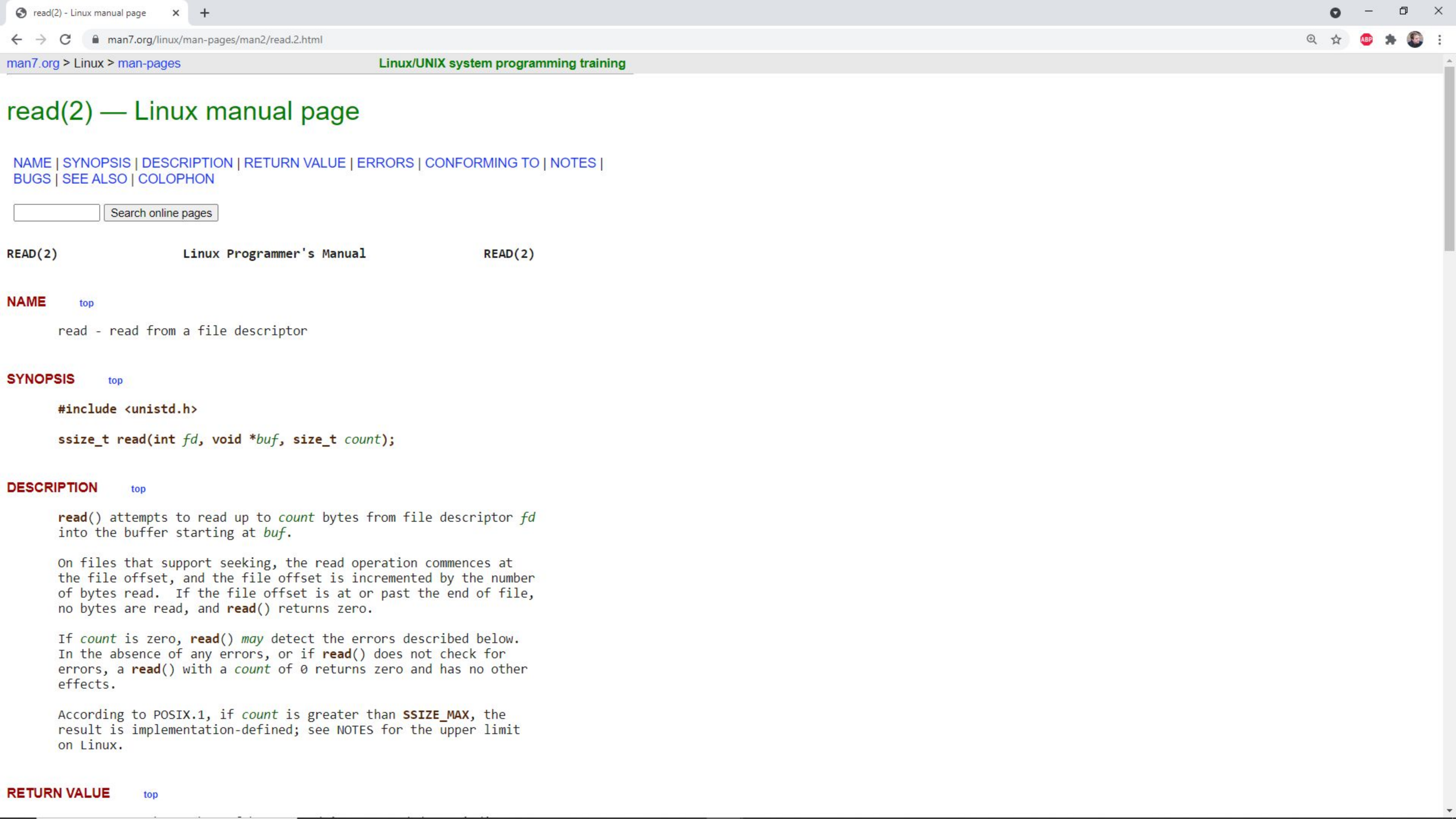
This interface is a member of the Java Collections Framework.

Since:

`1.2`

See Also:

`Collection, Set, ArrayList, LinkedList, Vector, Arrays.asList(Object[]), Collections.nCopies(int, Object), Collections.EMPTY_LIST, AbstractList, AbstractSequentialList`


read(2) - Linux manual page
man7.org/linux/man-pages/man2/read.2.html

man7.org > Linux > man-pages Linux/UNIX system programming training

read(2) — Linux manual page

NAME | SYNOPSIS | DESCRIPTION | RETURN VALUE | ERRORS | CONFORMING TO | NOTES | BUGS | SEE ALSO | COLOPHON

READ(2) Linux Programmer's Manual READ(2)

NAME top

read - read from a file descriptor

SYNOPSIS top

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

DESCRIPTION top

read() attempts to read up to *count* bytes from file descriptor *fd* into the buffer starting at *buf*.

On files that support seeking, the read operation commences at the file offset, and the file offset is incremented by the number of bytes read. If the file offset is at or past the end of file, no bytes are read, and **read**() returns zero.

If *count* is zero, **read**() *may* detect the errors described below. In the absence of any errors, or if **read**() does not check for errors, a **read**() with a *count* of 0 returns zero and has no other effects.

According to POSIX.1, if *count* is greater than **SSIZE_MAX**, the result is implementation-defined; see NOTES for the upper limit on Linux.

RETURN VALUE top

Web APIs

A “Web API” describes the functionality of software you access via a web interface.

There is NO standard way to write a Web API (yet).

Stripe API

https://stripe.com/docs/api/

GitHub API

https://docs.github.com/en/rest

National Weather Service API

https://www.weather.gov/documentation/services-web-api

Best Web API Design Practices:

1.

2.

3.

4.

5.

Clouds

Public Cloud

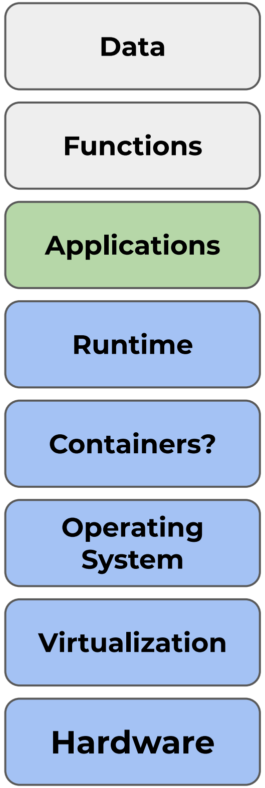
Private Cloud

Hybrid Cloud

EWS

linux.ews.illinois.edu

PaaS


Data

Functions

Applications

Runtime

Containers?

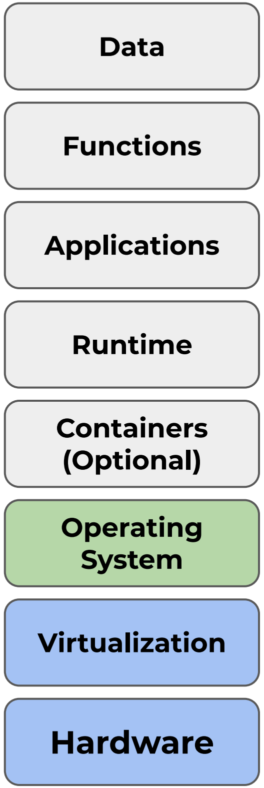
Operating System

Virtualization

Hardware

IaaS

https://vc.cs.illinois.edu/ui



Data

Functions

Applications

Runtime

Containers (Optional)

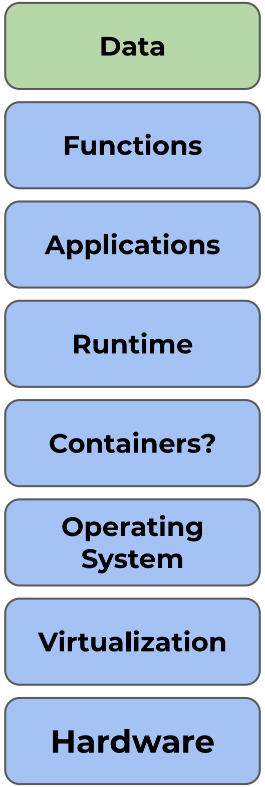
Operating System

Virtualization

Hardware

SaaS

https://queue.illinois.edu/



Data

Functions

Applications

Runtime

Containers?

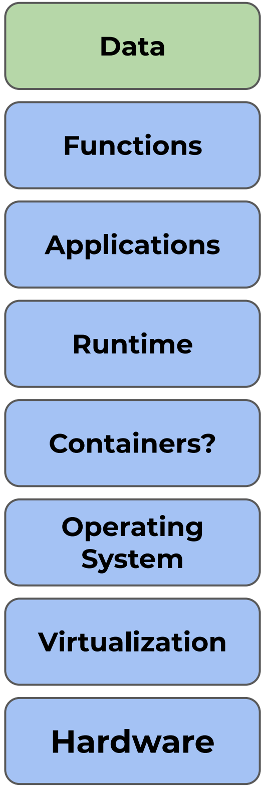
Operating System

Virtualization

Hardware

PrairieLearn

SaaS


Data

Functions

Applications

Runtime

Containers?

Operating System

Virtualization

Hardware