Serialization

`Serialization.serialize` — Function

```
serialize(stream::IO, value)
```

Write an arbitrary value to a stream in an opaque format, such that it can be read back by `deserialize`. The read-back value will be as identical as possible to the original. In general, this process will not work if the reading and writing are done by different versions of Julia, or an instance of Julia with a different system image. `Ptr` values are serialized as all-zero bit patterns (`NULL`).

An 8-byte identifying header is written to the stream first. To avoid writing the header, construct a `Serializer` and use it as the first argument to `serialize` instead. See also `Serialization.writeheader`.

```
serialize(filename::AbstractString, value)
```

Open a file and serialize the given value to it.

> **Julia 1.1**
>
> This method is available as of Julia 1.1.

`Serialization.deserialize` — Function

```
deserialize(stream)
```

Read a value written by `serialize`. `deserialize` assumes the binary data read from `stream` is correct and has been serialized by a compatible implementation of `serialize`. It has been designed with simplicity and performance as a goal and does not validate the data read. Malformed data can result in process termination. The caller has to ensure the integrity and

correctness of data read from `stream`.

```
deserialize(filename::AbstractString)
```

Open a file and deserialize its contents.

> **Julia 1.1**
>
> This method is available as of Julia 1.1.

`Serialization.writeheader` — Function

```
Serialization.writeheader(s::AbstractSerializer)
```

Write an identifying header to the specified serializer. The header consists of 8 bytes as follows:

Offset	Description
0	tag byte (0x37)
1-2	signature bytes "JL"
3	protocol version
4	bits 0-1: endianness: 0 = little, 1 = big
4	bits 2-3: platform: 0 = 32-bit, 1 = 64-bit
5-7	reserved

« SHA Shared Arrays »