Define *homoiconicity*.

Wikipedia: a property of some programming languages, in which the primary representation of programs is also a data structure in a primitive type of the language itself.

"code-as-data"

Describe the Clojure build process.

The reader converts the program text into forms which are then converted into Clojure data structures, which are then compiled.

What are the most common forms?

Boolean, number, string, character, keyword, symbol, nil, vector, list, map.

How are Clojure numbers different from Java's?

- Integers are automatically promoted to arbitrary-size `BigIntegers` as needed.
- `BigIntegers` and `BigDecimals` have a literal form, a number ending with `M`.

What can symbols refer to?

Functions with standard identifiers, operators, Java classes, Clojure namespaces, Java packages, data structures, refs

What are the rules for symbol names?

They may not begin with a number.
They consist of letters, numbers, `+`, `-`, `*`, `/`, `?`.

`.` and _ are also possible, but have special meaning with respect to namespaces.

How are string literals different from Java's?

- They can be multiline.
- They are displayed to the screen with escaped newlines.

How are strings different from Java's?

They are sequences of characters, so higher-order sequence functions work on them.

Describe `str`.

It creates a string, much like `toString`, but is n-ary and ignores `nil`.

How are escape characters created?

\backspace, \formfeed, \newline, \return, \space, \tab

How can a string be created from a sequence of characters?

```
(apply str [\a \b \c])
```

`apply` is making an n-ary call to `str` given a sequence argument.

How can you test for strict booleans?

Use `true?` and `false?`.

Give two ways to query a map.

- `(map key)`
- `(keyword-key map)`

Note the second form works only if the key is a keyword.

What are structs used for?

To document the fact that multiple maps are similar, i.e., they share common keys.

What special symbols are used in the documentation?

`&` indicates the following param is of variable arity and available as a seq.
`*` indicates the previous param is of variable arity.
`+` indicates the previous param is of variable, non-zero arity.
`?` indicates the previous param is optional.

Define a struct.

Instantiate a struct.

```
(defstruct name & keys)

(struct name & vals)
```

What is the basis of a struct?

The keys listed in the struct's definition.

How can one create a struct that lacks keys from the basis or has additional keys?

```
(struct-map name & inits)
```

Any missing keys will be given the value `nil` in the resulting struct.

How are structs different from maps?

The difference is mostly stylistic, although structs do store their values in indexed slots.

What's the difference between a reader macro and a form?

Reader macros are applied prior to the text being broken into forms.

What's the most common reader macro?

The comment, `;`.

Why are programs barred from defining additional reader macros?

To prevent code becoming unreadable to others, and to prevent Clojure from fragmenting into non-interoperable dialects.

How can you test for the type of a form?

Use the built in predicates `keyword?`, `symbol?`, etc.

What is the signature of `defn` with a single param list?

```
(defn name doc-string? attr-map? [params*] body)
```

What is the signature of `defn` with multiple param lists?

```
(defn name doc-string? attr-map?
 ([params*] body)+ )
```

What is `defn`, really?

A macro for:

```
(def name (fn [params*] exprs*))
```

The doc-string and attrs are added to the var metadata.

What is the vararg syntax?

Just as in the documentation syntax, us a `&` before the final parameter.

Give non-obvious reasons for using anonymous functions.

- To capture surrounding data in a function created at run-time (i.e., a closure) without giving many different functions the same name.
- To avoid a top-level binding for a function used exclusively inside another one.

What is the abbreviation for anonymous functions?

How is it implemented?

```
(# body)
```

where the arguments are given the names `%1`, `%2`, etc. `%` is the same as `%1`.

It's a reader macro.

What is a root binding?

The initial value of a var it was given in a `def`/`defn` statement.

How can a `var` be accessed directly, s
opposed to its value?

With the reader macro:

```
#'symbol
```

for `(var symbol)`.

What can vars do other than store values?

- Store metadata.
- Be rebound differently for each thread.
- Be aliased for unqualified use in other namespaces.

Which bindings are lexically scoped?

- parameters
- `(let [bindings*] exprs*)`

Where can destructuring be used?

What are the two kinds?

Destructuring can be used in fn parameter lists, `let`, and macros that expand to either of those.

Vectors can be put in the binding to capture sequential collections, or maps for associative collections.

How can you get a reference to the **entire** collection being destructured?

Use `:as name` in the destructuring. It will bind to the entire collection, not just the part being matched.

What makes a collection sequential?

It supports `nth`.

```
(coll index not-found?)
```

Name Clojure's special forms **not** for Java interop.

`def`, `if`, `do`, `let`, `quote`, `var`, `fn`, `loop`, `recur`, `throw`, `try`, `monitor-enter`, `monitor-exit`.

Name Clojure's special forms for Java interop.

., new, set!

Find the fully qualified name of a symbol.

Keep in mind ...

```
(resolve symbol)
```

... typically the symbol must be quoted to prevent evaluation into whatever it refers to.

How do you switch to a new namespce?

Create a new namespace?

`(in-ns name)`

switches to the namespace `name`, creating it if necessary.

What do all namespaces automatically include?

What does the `user` namespace bring in?

They include the package `java.lang`.

`user` also brings in `clojure.core`. Otherwise you would run:

```
(clojure.core/use 'clojure.core)
```

Avoid using the fully qualified names of Java classes.

```
(import '(package Class+))
```

Avoid using the fully qualified names of Clojure vars.

```
(use 'clojure.contrib.whatever)
(use '[clojure.contrib.whatever :only (func)])
```

What namespace are you currently in?

It's available as `*ns*`.

Where are imports from Clojure and Java typically done?

In the namespace declaration.

```
(ns name & references)
```

Each reference an be a `:use` or an `:import`.

What is the purpose of `do`?

It allows you to use multiple expressions where only one is allowed. The last expression is the the result of the whole expression. Previous expressions are executed for side-effects.

Describe `loop`.

It's just the same as `let`, except it introduces a recursion point which `recur` can return to.

```
(loop [bindings *] exprs*)
(recur exprs*)
```

Where you can you add metadata?

How do you do it?

Metadata can be added to collections and symbols.

```
(with-meta object metadata)
```

What is Clojure's semantic equality test?

Reference equality?

`=` is like Java's `equals`

`identical?` is like Java's ==

How is metadata usually accessed?

With the `^` macro.

`^x` is like `(meta x)`

Add keys to a map.

```
(assoc map k v & more-kvs)
```

Add `:tag` documentation metadata to a `defn`.

```
(defn #^{:tag BigInt} name [#^{:tag BigInt} i] body)
(defn #^BigInt name [#^BigInt i] body)
(defn name
  ([i] body)
  {:tag BigInt})
```

What is the difference between `with-meta` and the `#^` macro?

The reader macro adds metadata directly to a var or parameter.
`with-meta` adds it to the value.