

# Clojure



# Clojure

- ▶ A lisp

# Clojure

- ▶ A lisp
- ▶ That run on the JVM

# Clojure

- ▶ A lisp
- ▶ That run on the JVM
- ▶ Dynamic

# Clojure syntax

List

```
(1 "two" 3)
```

# Clojure syntax

List

```
(1 "two" 3)
```

Vector

```
[1 "two" 3]
```



# Clojure syntax

List

```
(1 "two" 3)
```

Vector

```
[1 "two" 3]
```

Map

```
{:first-key 1  
 :second-key "two"}
```

# Clojure syntax

List

```
(1 "two" 3)
```

Vector

```
[1 "two" 3]
```

Map

```
{:first-key 1  
 :second-key "two"}
```

Function definition

```
(defn my-function [param1 param2]  
  (str param1 param2))
```



# Clojure syntax

List

```
(1 "two" 3)
```

Vector

```
[1 "two" 3]
```

Map

```
{:first-key 1  
 :second-key "two"}
```

Function definition

```
(defn my-function [param1 param2]  
  (str param1 param2))
```

Function call

```
(my-function 1 "two") ;"1two"
```

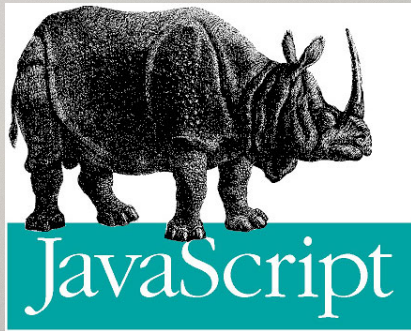
# Clojure syntax

Code is data, data is code

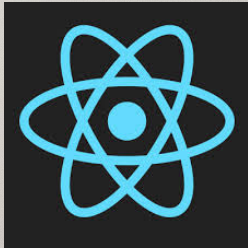
# Data structures

Immutable data structures

# Clojurescript



# React.js



# React.js

```
new Component(){  
    private String localState;  
  
    private void componentWillMount(){  
        //Setup  
    }  
    private void componentWillUnmount(){  
        //TearDown  
    }  
    private void renderState(){  
        return <input type="text" value={localState}/>;  
    }  
}
```



# core.async

```
(go (>! channel {:name "event" :value 3}))
```

```
(go (<! channel))
```

# core.async

- ▶ Event driven

# core.async

- ▶ Event driven
- ▶ Pub/Sub

# core.async

- ▶ Event driven
- ▶ Pub/Sub
- ▶ Multithreading

# core.async

Event driven

```
async_call(param, function(result){  
  console.log(result);  
});
```

# core.async

Event driven

```
async_call(param, function(result){  
  console.log(result);  
});
```

```
async_call(param, new Future());
```



# core.async

Event driven

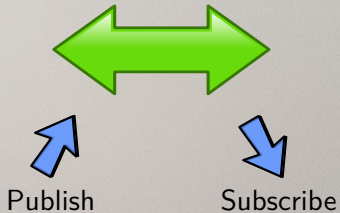
```
async_call(param, function(result){  
  console.log(result);  
});
```

```
async_call(param, new Future());
```

```
async_call(param, channel);
```

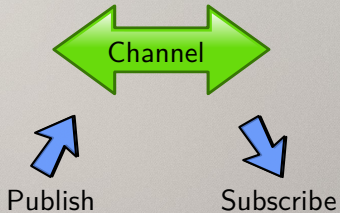
# core.async

Pub/Sub



# core.async

Pub/Sub



# Next

- ▶ Web server

# Next

- ▶ Web server
- ▶ `core.typed`

# Next

- ▶ Web server
- ▶ `core.type`
- ▶ Datomic?