

COMP 322: Fundamentals of Parallel Programming

Lecture 10: Event-Based Programming

Mack Joyner
mjoyner@rice.edu

<http://comp322.rice.edu>



What is an Event-Based Programming?

- Event-based programming is a paradigm where actions are performed (event handlers) in response to events.
- Events are often triggered by a user (GUI, web programming)
- Events include:
 - Mouse events (clicks, mouse over)
 - Timeouts, Intervals
 - Keyboard events (key press down/up)

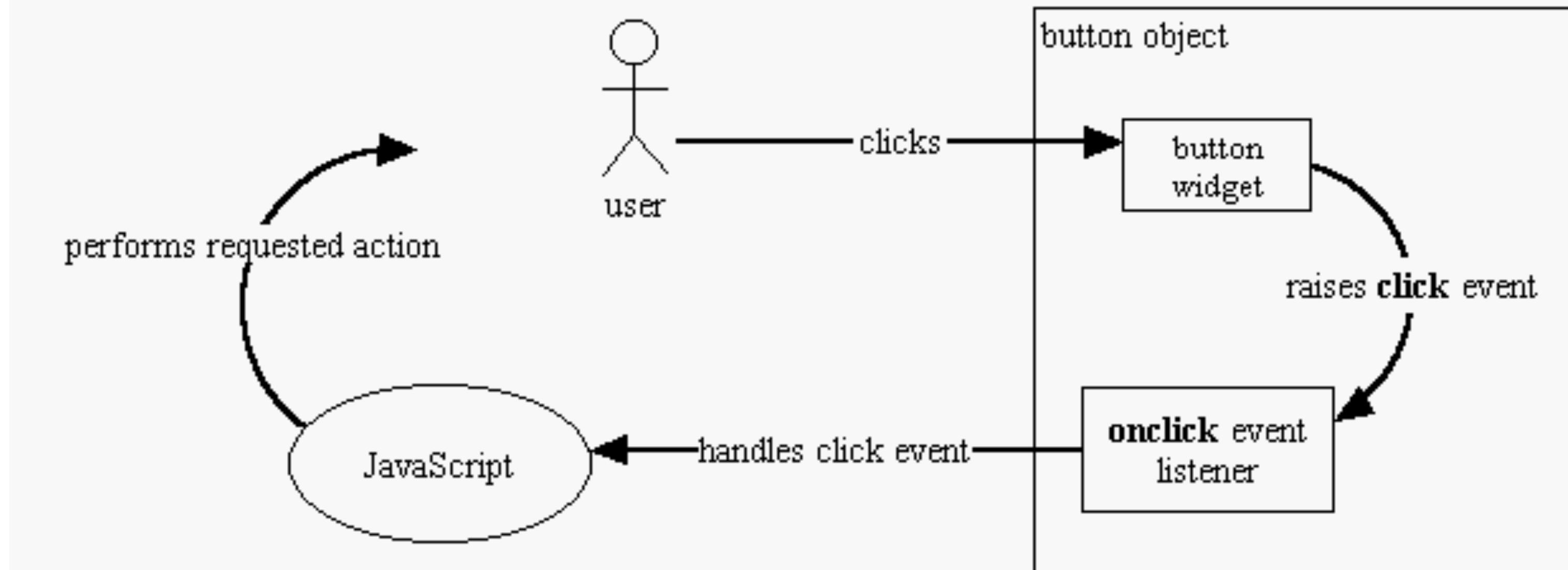
See: https://en.wikipedia.org/wiki/Event-driven_programming



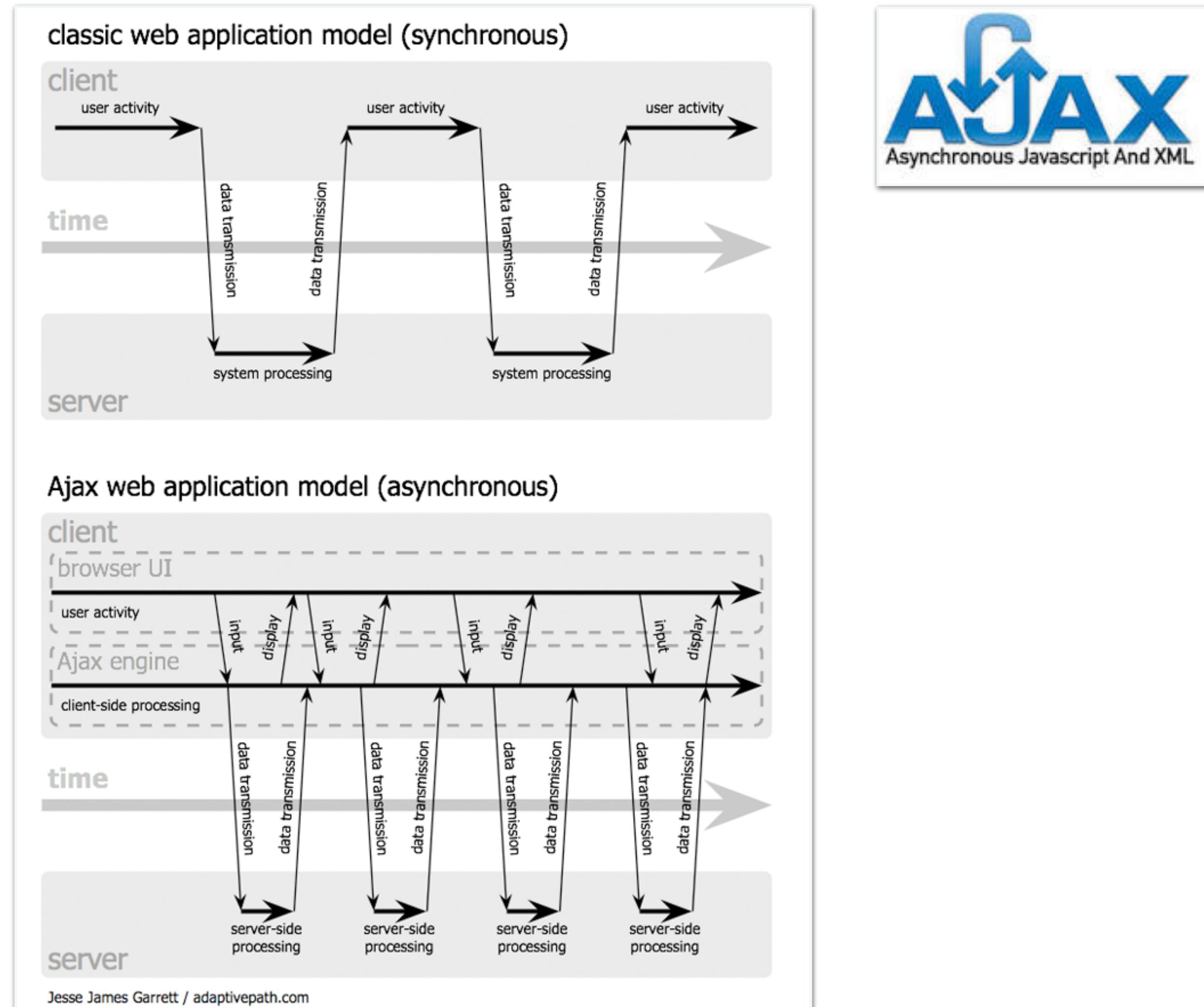
Event Handling

```
<button value="Click Me" onclick="alert('Thank you') " />
```

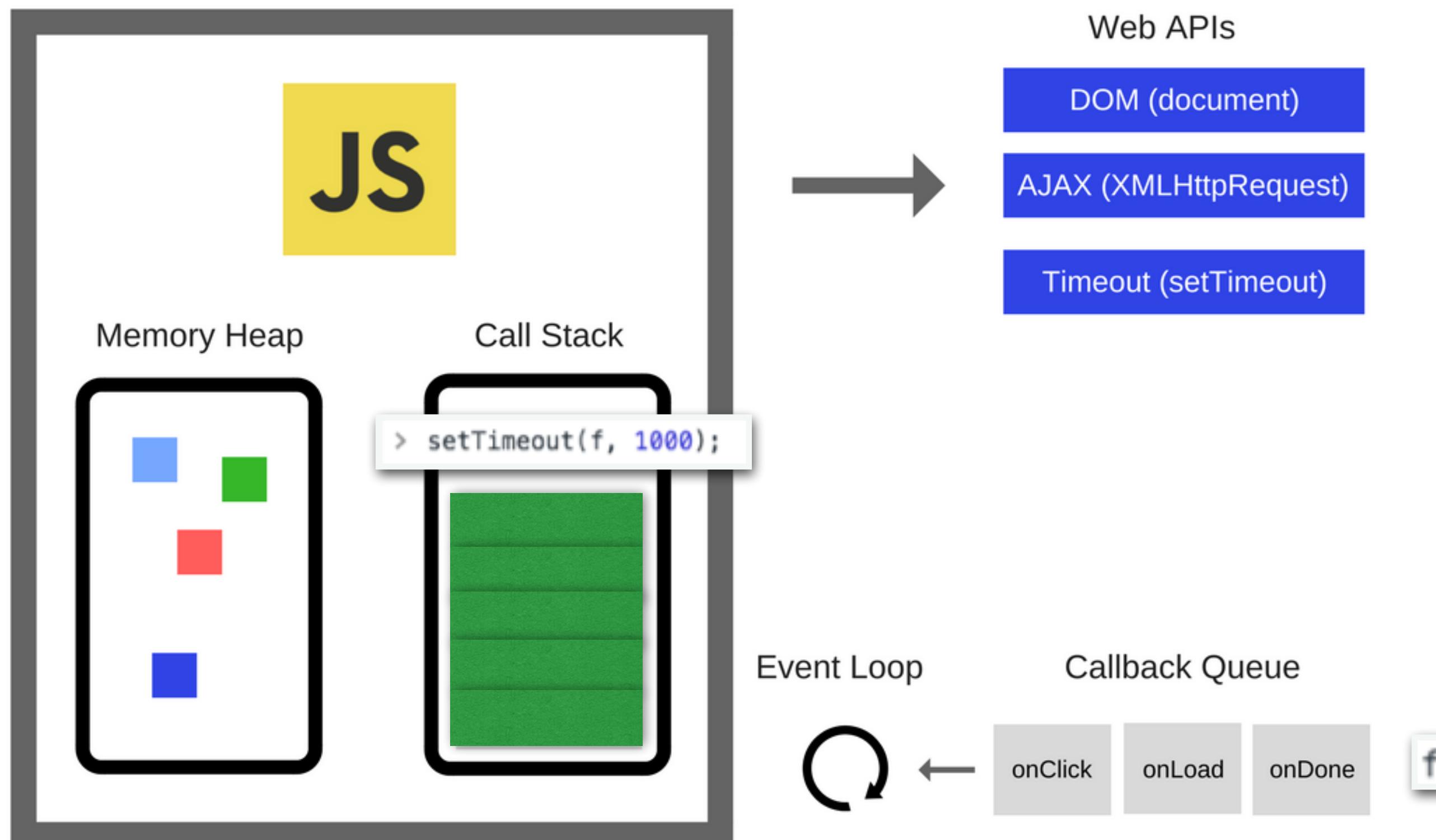
JavaScript



Asynchronous Event Handling



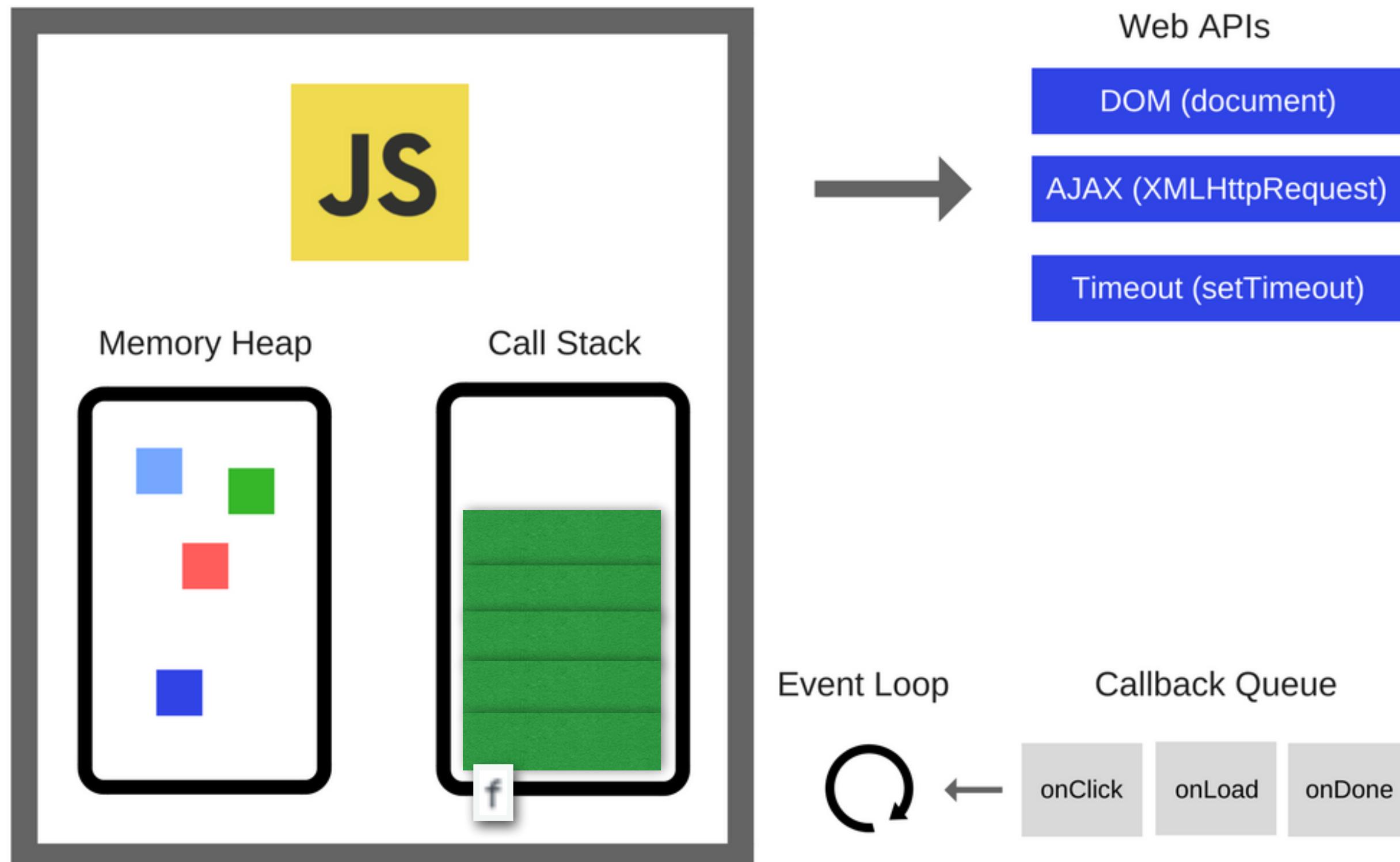
System Control over Event Response



Source: <https://blog.sessionstack.com/how-does-javascript-actually-work-part-1-b0bacc073cf>



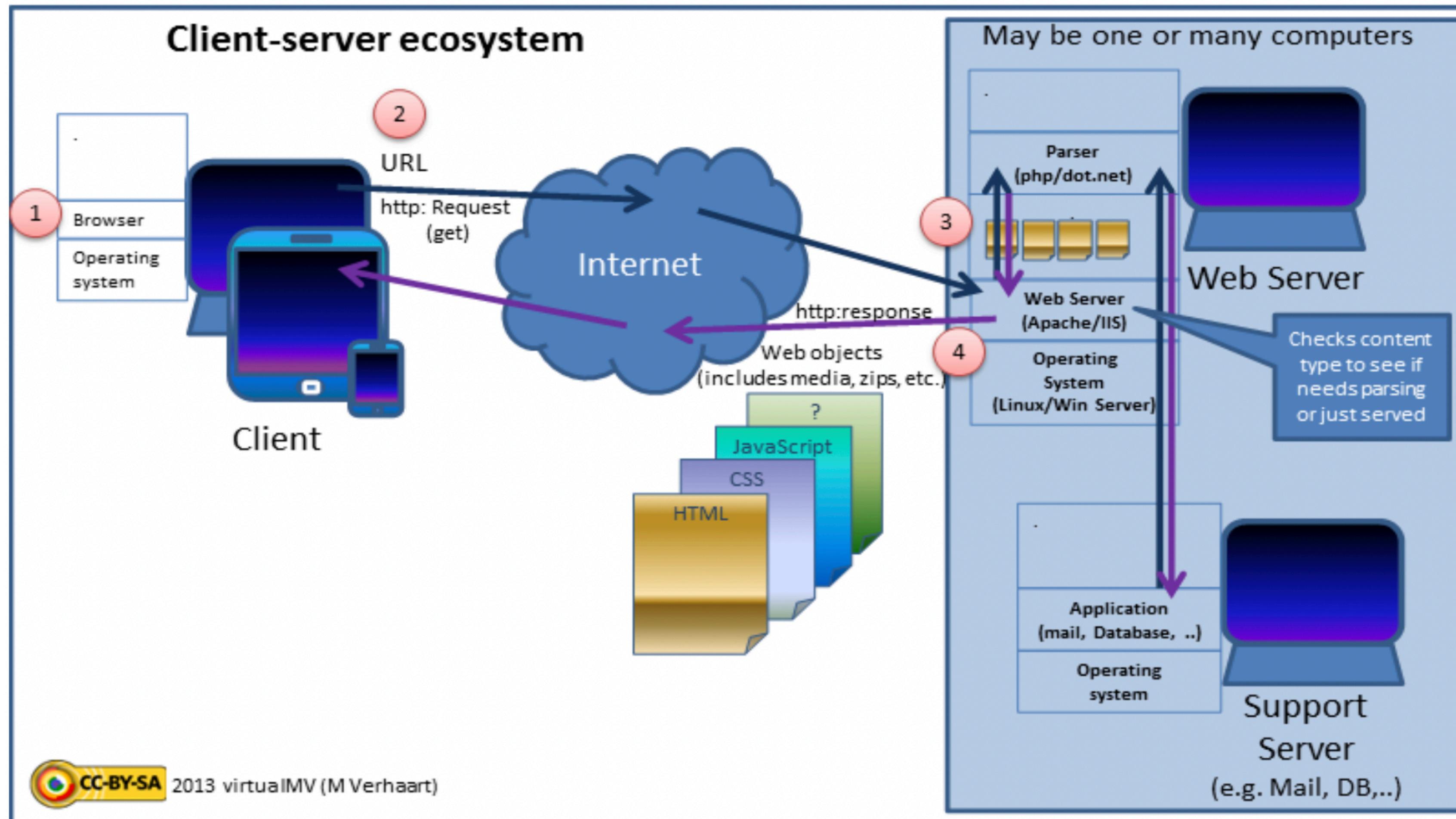
System Control over Event Response



Would the execution order change if `setTimeOut` used 0 instead of 1000?



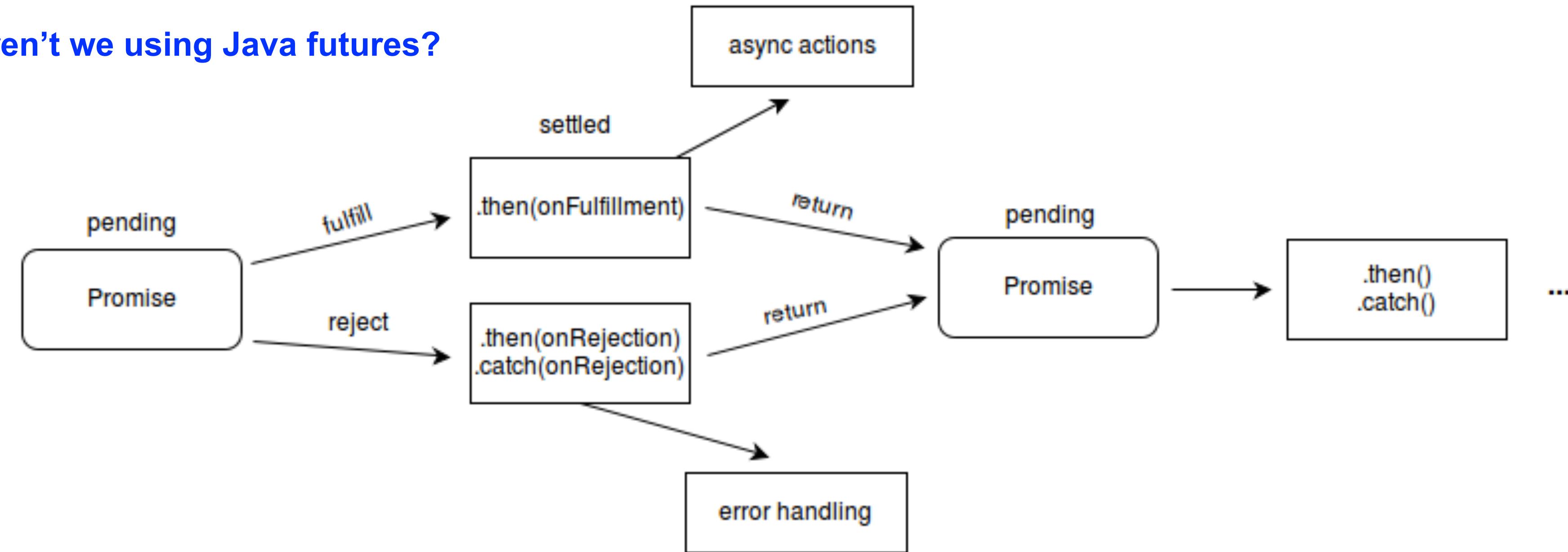
World Wide Web



JavaScript Promises

Java has both Futures (since JDK 1.5) and Promises (CompletableFuture since JDK 8)

Wait...then why aren't we using Java futures?



https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

See also <http://www.html5rocks.com/en/tutorials/es6/promises/>



Promise: Fetching Web Data

Inline function callbacks
with arrow notation

Same as:
.then(function(r) {
 return r.json()
})
r.json() returns a Promise, the next
then() is called when json() resolves.

```
> fetch
< f fetch() { [native code]
> fetch('https://jsonplaceholder.typicode.com/posts').then(r => r.json())
< ▶ Promise {<pending>} 1
  ▶ __proto__: Promise
    [[PromiseStatus]]: "resolved"
  ▶ [[PromiseValue]]: Array(100)
    ▶ 0: {userId: 1, id: 1, title: "sunt aut facere repellat provident occaecati", body: "quia et sed quo", category: "quia et sed quo", type: "quia et sed quo", status: "quia et sed quo", priority: "quia et sed quo", created_at: "quia et sed quo", updated_at: "quia et sed quo", deleted_at: null}
    ▶ 1: {userId: 1, id: 2, title: "qui est esse", body: "est rerum tempore vitae", category: "qui est esse", type: "qui est esse", status: "qui est esse", priority: "qui est esse", created_at: "qui est esse", updated_at: "qui est esse", deleted_at: null}
    ▶ 2: {userId: 1, id: 3, title: "ea molestias quasi exercitationem repellat quia et sed quo", body: "et ea vero quia laudantium autem", category: "ea molestias quasi exercitationem repellat quia et sed quo", type: "ea molestias quasi exercitationem repellat quia et sed quo", status: "ea molestias quasi exercitationem repellat quia et sed quo", priority: "ea molestias quasi exercitationem repellat quia et sed quo", created_at: "ea molestias quasi exercitationem repellat quia et sed quo", updated_at: "ea molestias quasi exercitationem repellat quia et sed quo", deleted_at: null}
    ▶ 3: {userId: 1, id: 4, title: "eum et est occaecati", body: "ullam et saepe", category: "eum et est occaecati", type: "eum et est occaecati", status: "eum et est occaecati", priority: "eum et est occaecati", created_at: "eum et est occaecati", updated_at: "eum et est occaecati", deleted_at: null}
    ▶ 4: {userId: 1, id: 5, title: "nesciunt quas odio", body: "repudiandae veniam", category: "nesciunt quas odio", type: "nesciunt quas odio", status: "nesciunt quas odio", priority: "nesciunt quas odio", created_at: "nesciunt quas odio", updated_at: "nesciunt quas odio", deleted_at: null}
    ▶ 5: {userId: 1, id: 6, title: "dolorem eum magni eos aperiam quia", body: "dolore placeat", category: "dolorem eum magni eos aperiam quia", type: "dolorem eum magni eos aperiam quia", status: "dolorem eum magni eos aperiam quia", priority: "dolorem eum magni eos aperiam quia", created_at: "dolorem eum magni eos aperiam quia", updated_at: "dolorem eum magni eos aperiam quia", deleted_at: null}
    ▶ 6: {userId: 1, id: 7, title: "magnam facilis autem", body: "dolore placeat", category: "magnam facilis autem", type: "magnam facilis autem", status: "magnam facilis autem", priority: "magnam facilis autem", created_at: "magnam facilis autem", updated_at: "magnam facilis autem", deleted_at: null}
    ▶ 7: {userId: 1, id: 8, title: "dolorem dolore est ipsam", body: "dignissimos", category: "dolorem dolore est ipsam", type: "dolorem dolore est ipsam", status: "dolorem dolore est ipsam", priority: "dolorem dolore est ipsam", created_at: "dolorem dolore est ipsam", updated_at: "dolorem dolore est ipsam", deleted_at: null}
    ▶ 8: {userId: 1, id: 9, title: "nesciunt iure omnis dolorem tempora et accusamus", body: "et ea vero quia laudantium autem", category: "nesciunt iure omnis dolorem tempora et accusamus", type: "nesciunt iure omnis dolorem tempora et accusamus", status: "nesciunt iure omnis dolorem tempora et accusamus", priority: "nesciunt iure omnis dolorem tempora et accusamus", created_at: "nesciunt iure omnis dolorem tempora et accusamus", updated_at: "nesciunt iure omnis dolorem tempora et accusamus", deleted_at: null}
    ▶ 9: {userId: 1, id: 10, title: "optio molestias id quia eum", body: "et ea vero quia laudantium autem", category: "optio molestias id quia eum", type: "optio molestias id quia eum", status: "optio molestias id quia eum", priority: "optio molestias id quia eum", created_at: "optio molestias id quia eum", updated_at: "optio molestias id quia eum", deleted_at: null}
    ▶ 10: {userId: 2, id: 11, title: "et ea vero quia laudantium autem", body: "et ea vero quia laudantium autem", category: "et ea vero quia laudantium autem", type: "et ea vero quia laudantium autem", status: "et ea vero quia laudantium autem", priority: "et ea vero quia laudantium autem", created_at: "et ea vero quia laudantium autem", updated_at: "et ea vero quia laudantium autem", deleted_at: null}
    ▶ 11: {userId: 2, id: 12, title: "in quibusdam tempore odit est dolorem", body: "et ea vero quia laudantium autem", category: "in quibusdam tempore odit est dolorem", type: "in quibusdam tempore odit est dolorem", status: "in quibusdam tempore odit est dolorem", priority: "in quibusdam tempore odit est dolorem", created_at: "in quibusdam tempore odit est dolorem", updated_at: "in quibusdam tempore odit est dolorem", deleted_at: null}
    ▶ 12: {userId: 2, id: 13, title: "dolorum ut in voluptas mollitia et saepe", body: "et ea vero quia laudantium autem", category: "dolorum ut in voluptas mollitia et saepe", type: "dolorum ut in voluptas mollitia et saepe", status: "dolorum ut in voluptas mollitia et saepe", priority: "dolorum ut in voluptas mollitia et saepe", created_at: "dolorum ut in voluptas mollitia et saepe", updated_at: "dolorum ut in voluptas mollitia et saepe", deleted_at: null}
    ▶ 13: {userId: 2, id: 14, title: "voluptatem eligendi optio", body: "fuga et", category: "voluptatem eligendi optio", type: "voluptatem eligendi optio", status: "voluptatem eligendi optio", priority: "voluptatem eligendi optio", created_at: "voluptatem eligendi optio", updated_at: "voluptatem eligendi optio", deleted_at: null}
    ▶ 14: {userId: 2, id: 15, title: "eveniet quod temporibus", body: "reprehenderit", category: "eveniet quod temporibus", type: "eveniet quod temporibus", status: "eveniet quod temporibus", priority: "eveniet quod temporibus", created_at: "eveniet quod temporibus", updated_at: "eveniet quod temporibus", deleted_at: null}
    ▶ 15: {userId: 2, id: 16, title: "sint suscipit perspiciatis velit dolorum re", body: "et ea vero quia laudantium autem", category: "sint suscipit perspiciatis velit dolorum re", type: "sint suscipit perspiciatis velit dolorum re", status: "sint suscipit perspiciatis velit dolorum re", priority: "sint suscipit perspiciatis velit dolorum re", created_at: "sint suscipit perspiciatis velit dolorum re", updated_at: "sint suscipit perspiciatis velit dolorum re", deleted_at: null}
    ▶ 16: {userId: 2, id: 17, title: "fugit voluptas sed molestias voluptatem pro", body: "et ea vero quia laudantium autem", category: "fugit voluptas sed molestias voluptatem pro", type: "fugit voluptas sed molestias voluptatem pro", status: "fugit voluptas sed molestias voluptatem pro", priority: "fugit voluptas sed molestias voluptatem pro", created_at: "fugit voluptas sed molestias voluptatem pro", updated_at: "fugit voluptas sed molestias voluptatem pro", deleted_at: null}
    ▶ 17: {userId: 2, id: 18, title: "voluptate et itaque vero tempora molestiae", body: "et ea vero quia laudantium autem", category: "voluptate et itaque vero tempora molestiae", type: "voluptate et itaque vero tempora molestiae", status: "voluptate et itaque vero tempora molestiae", priority: "voluptate et itaque vero tempora molestiae", created_at: "voluptate et itaque vero tempora molestiae", updated_at: "voluptate et itaque vero tempora molestiae", deleted_at: null}
    ▶ 18: {userId: 2, id: 19, title: "adipisci placeat illum aut reiciendis qui", body: "et ea vero quia laudantium autem", category: "adipisci placeat illum aut reiciendis qui", type: "adipisci placeat illum aut reiciendis qui", status: "adipisci placeat illum aut reiciendis qui", priority: "adipisci placeat illum aut reiciendis qui", created_at: "adipisci placeat illum aut reiciendis qui", updated_at: "adipisci placeat illum aut reiciendis qui", deleted_at: null}
    ▶ 19: {userId: 2, id: 20, title: "doloribus ad provident suscipit at", body: "et ea vero quia laudantium autem", category: "doloribus ad provident suscipit at", type: "doloribus ad provident suscipit at", status: "doloribus ad provident suscipit at", priority: "doloribus ad provident suscipit at", created_at: "doloribus ad provident suscipit at", updated_at: "doloribus ad provident suscipit at", deleted_at: null}
    ▶ 20: {userId: 3, id: 21, title: "asperiores ea ipsam voluptatibus modi minim", body: "et ea vero quia laudantium autem", category: "asperiores ea ipsam voluptatibus modi minim", type: "asperiores ea ipsam voluptatibus modi minim", status: "asperiores ea ipsam voluptatibus modi minim", priority: "asperiores ea ipsam voluptatibus modi minim", created_at: "asperiores ea ipsam voluptatibus modi minim", updated_at: "asperiores ea ipsam voluptatibus modi minim", deleted_at: null}
```

JavaScript Async/Await (like HJ Futures/Data Driven Tasks)

- **Async** functions always return Promise
- **Await** can only be inside async
- Expression after **await** is like using Promise **then**

```
(async () => {
  const connector = mongoose.connect(connectionString)
  const username = process.argv[2].split('=')[1]

  let user = await connector.then(async () => {
    return findUser(username)
  })

  if (!user) {
    user = await createUser(username)
  }

  console.log(user)
  process.exit(0)
})()
```

Need response before sending result



Login/Logout registered users with Futures

```
var username = ...
var password = ...
...
var regUser = future(() -> registerNewUser(username, password)); // { username: user, result: "success" or "failure"}
...
var logUser = future(() -> loginUser(username, password)); // {userId: id, result: "success" or "failure" }
...
var loggedIn = future(() -> isLoggedIn(id)); // { userId: id, result: "success" or "failure" }
...
var logOut = future(() -> logoutUser(id)); // { result: "success" or "failure" }
...
```

What future dependencies are missing?



Login/Logout registered users with Futures

```
var username = ...
var password = ...
...
var regUser = future(() -> registerNewUser(username, password)); // { username: user, result: "success" or "failure"}
...
var logUser = future(() -> { if (regUser.get().result.equals("success"))
    return loginUser(username, password); // {userId: id, result: "success" or "failure"}
    return {result: "failure"};
});
...
var loggedIn = future(() -> { if (logUser.get().result.equals("success"))
    return isLoggedIn(logUser.get().userId); // {userId: id, result: "success" or "failure" }
    return {result: "failure"};
});
...
var logOut = future(() -> { if (loggedIn.get().result.equals("success"))
    return logoutUser(loggedIn.get().userId)); // { result: "success" or "failure" }
    return {result: "failure"};
});
```

What are the problems here? Should we use DDF/DDTs?



Login/Logout registered users with DDTs

```
var username = ...
var password = ...
...
var regUser = newDataDrivenFuture();
var logUser = newDataDrivenFuture();
var loggedIn = newDataDrivenFuture();
var logOut = newDataDrivenFuture();

...
async(() -> regUser.put(registerNewUser(username, password))); // { username: user, result: "success" or "failure"}
...
asyncAwait(regUser, () -> { if (regUser.safeGet().result.equals("success"))
    logUser.put(loginUser(username, password)); // {userId: id, result: "success" or "failure"}
    else
        logUser.put({result: "failure"});    });
    ...
asyncAwait(logUser, () -> { if (logUser.safeGet().result.equals("success"))
    loggedIn.put(isLoggedIn(logUser.safeGet().userId)); // {userId: id, result: "success" or "failure" }
    else
        loggedIn.put({result: "failure"});    });
    ...
asyncAwait(loggedIn, () -> { if (loggedIn.safeGet().result.equals("success"))
    logOut.put(logoutUser(loggedIn.safeGet().userId)); // { result: "success" or "failure" }
    else
        logOut.put({result: "failure"});    });
    ...
}
```

