



architecture best practices



things to consider

data model
application structure
browser history
simplifying your server
authentication



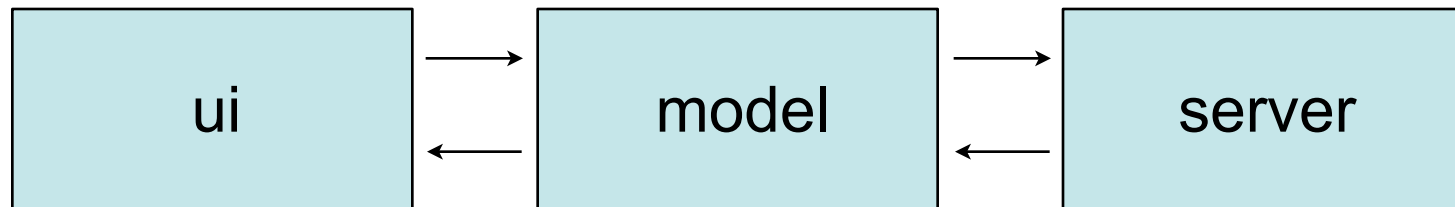
old-school ajax

Wow, we can update something without a refresh?
That's cool enough!

Just refetch data whenever you feel like it.



client-side data model





model

view

controller



dependency injection

```
DataModel model;
DataView myView = new DataView(model);
...
public DataView(DataModel model) {
    this.model = model;
    model.addChangeListener(this);
}
```



client-server communication

never forget you're writing a distributed application!

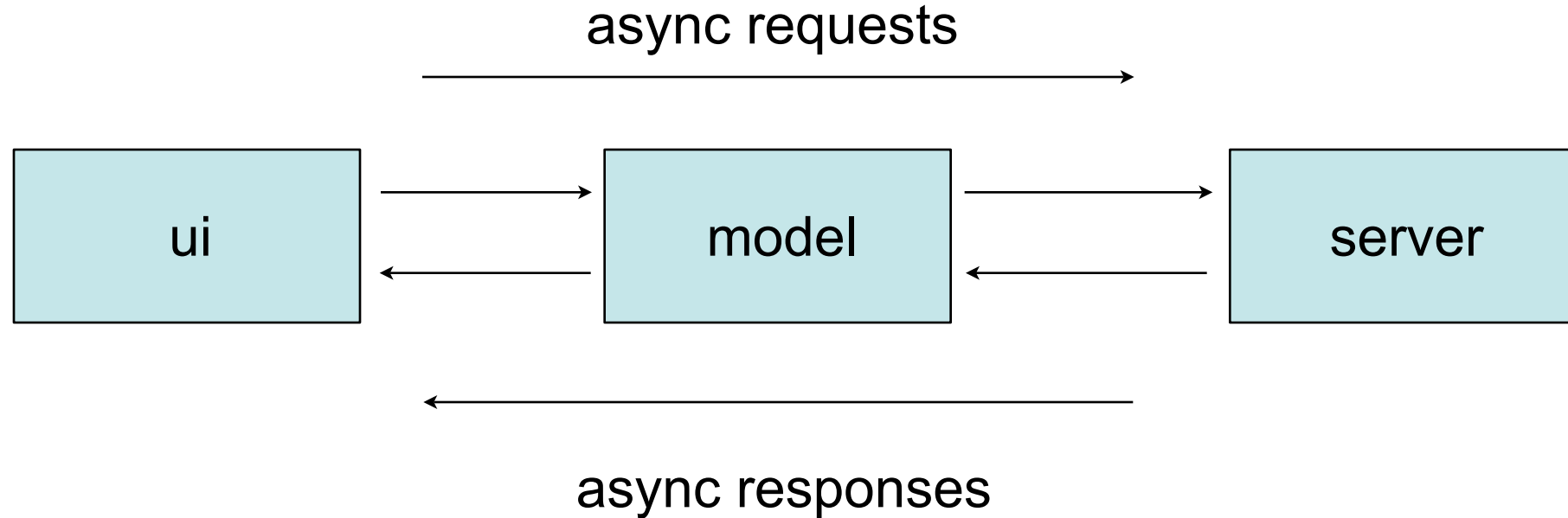


embrace asynchronicity

```
myInterface.getSomeData(query, new
    AsyncCallback<MyData>() {
        public void onSuccess(MyData result) {
            // do something interesting
        }
        public void onFailure(Throwable caught) {
            // warn the user or retry
        }
    });
```



model updates will be async



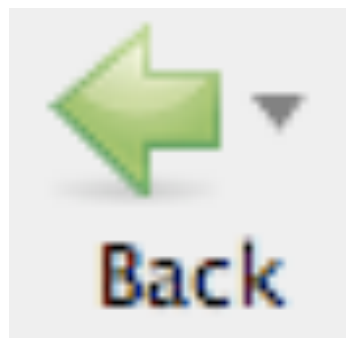
application structure

browser history
large-scale structure





history



define “location” for your app

<http://mail.google.com/mail/#inbox>



history in gwt

```

History.addHistoryListener(this);
...
public void onUserAction() {
    History.newItem(createTokenFromState());
}
...
public void onHistoryChanged(String historyToken) {
    updateAppState(parseToken(historyToken));
}
  
```



bookmarks for free!



breaking up large apps

multiple modules on separate pages



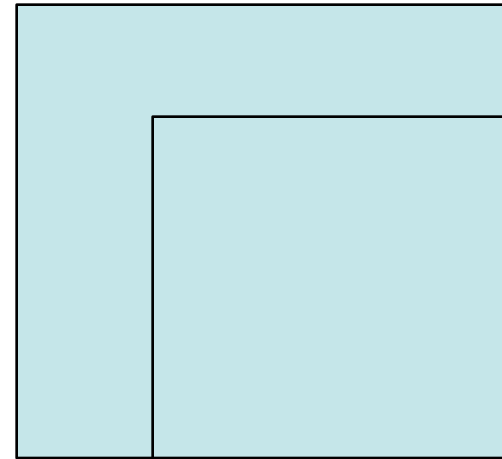
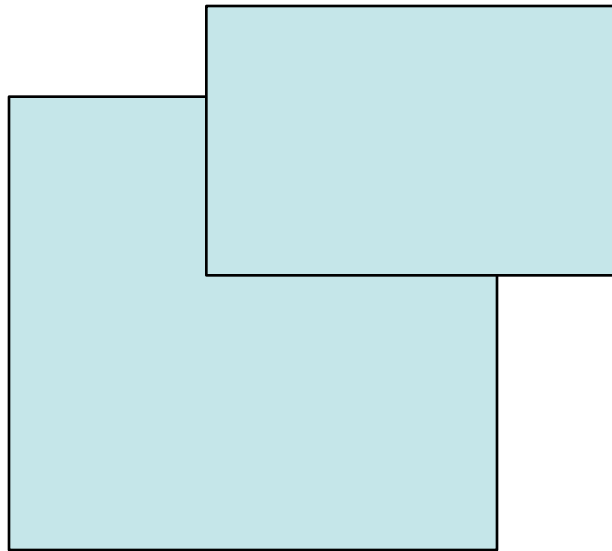
work single page into history stack

```
new Frame(GWT.getModuleBaseURL() +  
          "/settings.html");
```

```
Window.open(GWT.getModuleBaseURL() +  
            "/settings.html",  
            "_blank", "");
```

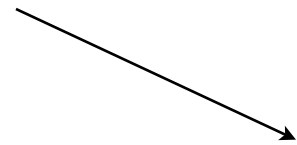


use multiple windows or iframes

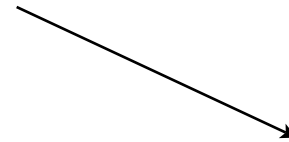


simplifying your server

client-side state



stateless server



scalability



authentication

can use cookies

or just use authentication tokens



```
public interface MyService {  
    // returns authorization token  
    String login(String user, String pass);  
    Data getSensitiveData(String authToken);  
}
```





questions?

