CS 349: Algorithms

Michael Noukhovitch

Winter 2015, University of Waterloo

Notes written from Michael Terry's lectures.

Contents

1 Introduction

1.1 Definitions

Interface: external presentation to user

- **controls**: manipulated to communicate intent
- **presentation**: what communicates response

Interaction: the actions a user must do to elicit corresponding response

1. action and dialog
2. unfolds over time

2 Events

2.1 Event Loop

```
while(true) {
    if there is an event on queue:
        dequeue it
        dispatch it
}
```

2.2 Timer

Some events are triggered by a timer, if that event's execution time is longer than the timer interval then by the end of the event execution, you should add another of your event to the queue!

2.3 Interactor Tree

We need a way to send information about what object is clicked
interactor tree: hierarchical tree-based organization of widgets

- each component's location is specified relative to parent
- we use **containers** whose sole purpose is to contain components
- events go **down** the tree to **capture** the target clicked
- event bubble **up** the tree to **handle** an event (e.g. EventListener)

2.4 Event Propogation

when an event happens:

1. calculate the parent node path
2. loop through it and execute capture phase handlers

3. execute DOM level 1 phase handler
4. execute bubble phase handlers
5. execute default browser behaviour

3 Model View Controller

3.1 Idea

We decouple presentation from data using the **observer** design pattern. This separation allots benefits:

- **change the UI**: easy to change how we interact with data
- **multiple view**: have different views of same data
- **code reuse**: different logic for same view etc..
- **testing**: data separation allows better logic testing

3.2 Description

Model: manages the data

- represent the data
- methods to manipulate data
- create and notify listeners

View: manages the presentation

- renders the data in a model
- references to the model
- is a listener to the model

Controller: manages user interaction

- between the model and view
- helps interpret input and model events

4 Layout

4.1 Layout Manager

Layout Manager: keeps the layout for components given their constraints and preferences

- uses composite and strategy design pattern

4.1.1 Dynamic Layout

Dynamic Layout: maintain consistency with spatial layout

- reallocate space for widget
- adjust location and size
- change visibility, look, feel

4.1.2 Layout Strategies

- **fixed layout**
- **intrinsic size**: find each item's preferred size and the container will grow to perfectly contain each item
- **variable intrinsic size**: layout determined in bottom-up and top-down phases
- **struts and spring**: items can either be fixed (strut) or variable (spring)

4.2 Responsive Design

Responsive Design: change layout to adapt to screen sizes of different devices

4.2.1 CSS

CSS: specifying formatting

- consistency
- reduce size (cache CSS)
- code reuse
- separation of concerns

CSS reset: normalize appearance across browsers

4.2.2 Cascade

Layout resolves CSS rules and renders following these rules:

1. find all declarations that match the element
2. sort declarations by `!important`
3. sort by origin (author > web browser)
4. sort by specificity of selector
5. sort by order (later rule wins)