Informatics 134

Software User Interfaces
Spring 2024

Mark S. Baldwin
baldwinm@ics.uci.edu

4/11/2024

Agenda

1. Upcoming
2. Callbacks and Events
3. Demo
4. References

Upcoming

Upcoming

Agenda

- Today:
    - Events and Callbacks
    - Demo
- Next Week:
    - A2 Due Tuesday (4/23)
    - A3 Due Tuesday (4/23)

Callbacks and Events

Event Callbacks

User input, events, and action

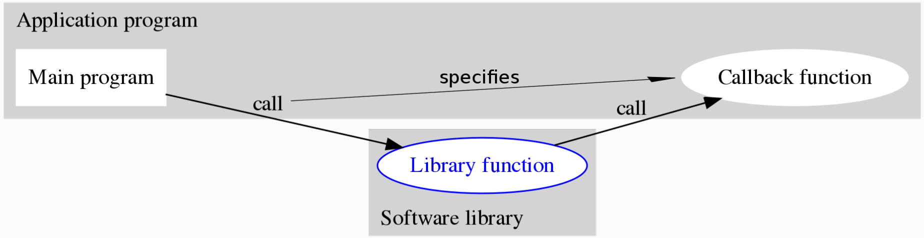
- Differ across programming languages and toolkits
- Referred to as: “callbacks”, “event handlers”, “actions”, and others

Event Callbacks

User input, events, and action

In the browser...

- Input widgets (text, check, button, heading, div, etc)
- Angular’s data bindings (e.g., “ng-bind”)



[Wikipedia, 2021]

An HTML and Javascript Example

A function is referenced via HTML attribute and *called* when the specified event is performed (a click).

Here the DOM manages the connections for us, but in our Toolkit we are responsible for connecting user input to action.

```
function buttonClick(e){
    ...do some action
}
```

```
<button onclick="buttonClick(this)">...</button>
```

User input, events, and action

In Javascript/Typescript and SVG.js

- Callbacks are functions that we pass as objects
- First, we must make the function anonymous
- And because functions are first-class citizens in JavaScript...?

```
function buttonClick(e){
     ...do some action
}

// becomes:

var buttonClick = function(e){
     ...do some action
}
```

User input, events, and action

In Javascript/Typescript and SVG.js

- We can pass them to other functions as arguments. So...
- Callbacks are simply functions that we pass as objects
- We can use callbacks to customize the propagation of user input actions

```
var buttonClick = function(e){
    console.log(e)
}

function MyCallback(action){
    action("MyCallback was called")
}

MyCallback(buttonClick)
```

```
> "MyCallback was called"
```

User input, events, and action

In Javascript/Typescript and SVG.js

> In the example to the right, what will be the output of a hover event?

```
var w = new MyWidget();
let callback = function(event:any){
    console.log("I am being hovered!");
};
w.onHover(callback);

```

```
class MyWidget extends Widget{
    onHover(callback: { (event?:any): void }):void{
        this.attach(callback);
    }
    ...
    hoverState(): void{
        this.raise(new EventArgs(this));
    }
}
```

Event Callbacks

User input, events, and action

Why? How do callbacks help us build toolkits?

- Separation of concerns
- Clean up operations
- Pass control to consuming (or calling) code
- Asynchronous operations (promises in JS)

Demo

Let’s Dive In!

References

References i

Wikipedia (2021).
Callback (computer programming).