Structural Design Patterns

Patterns and Antipatterns in Javascript

Chrysa Papadaki // chr.papadaki@tum.de

Nishant Gupta // nishant.gupta@tum.de

Master Seminar | Technische Universität München | Garching, 29.09.2015

Outcome

- Understand the nature and benefits of Structural Design Patterns

- Learn how to utilize some of the most popular:
  - ★ Proxy
  - ★ Decorator
  - ★ Facade
  - ★ Composite

Structural Design Patterns

- ★ Focus on the relationship between objects and their inheritance to create the wider application
- ★ Each one has a different purpose
- ★ Similar to the simpler concept of *data structures*

Benefits

Structural Design Patterns - Benefits I

★ **Share data between objects**

Flyweight

Structural Design Patterns - Benefits II

- ★ Share data between objects
- ★ **Treat complex and primitive objects uniformly**

  Composite

Structural Design Patterns - Benefits III

- ★ Share data between objects
- ★ Treat complex and primitive objects uniformly
- ★ **Add additional functionality to an object at runtime**

  Decorator

Structural Design Patterns - Benefits IV

- ★ Share data between objects
- ★ Treat complex and primitive objects uniformly
- ★ Add additional functionality to an object at runtime
- ★ **Hide complex code/interface behind a more simplified**

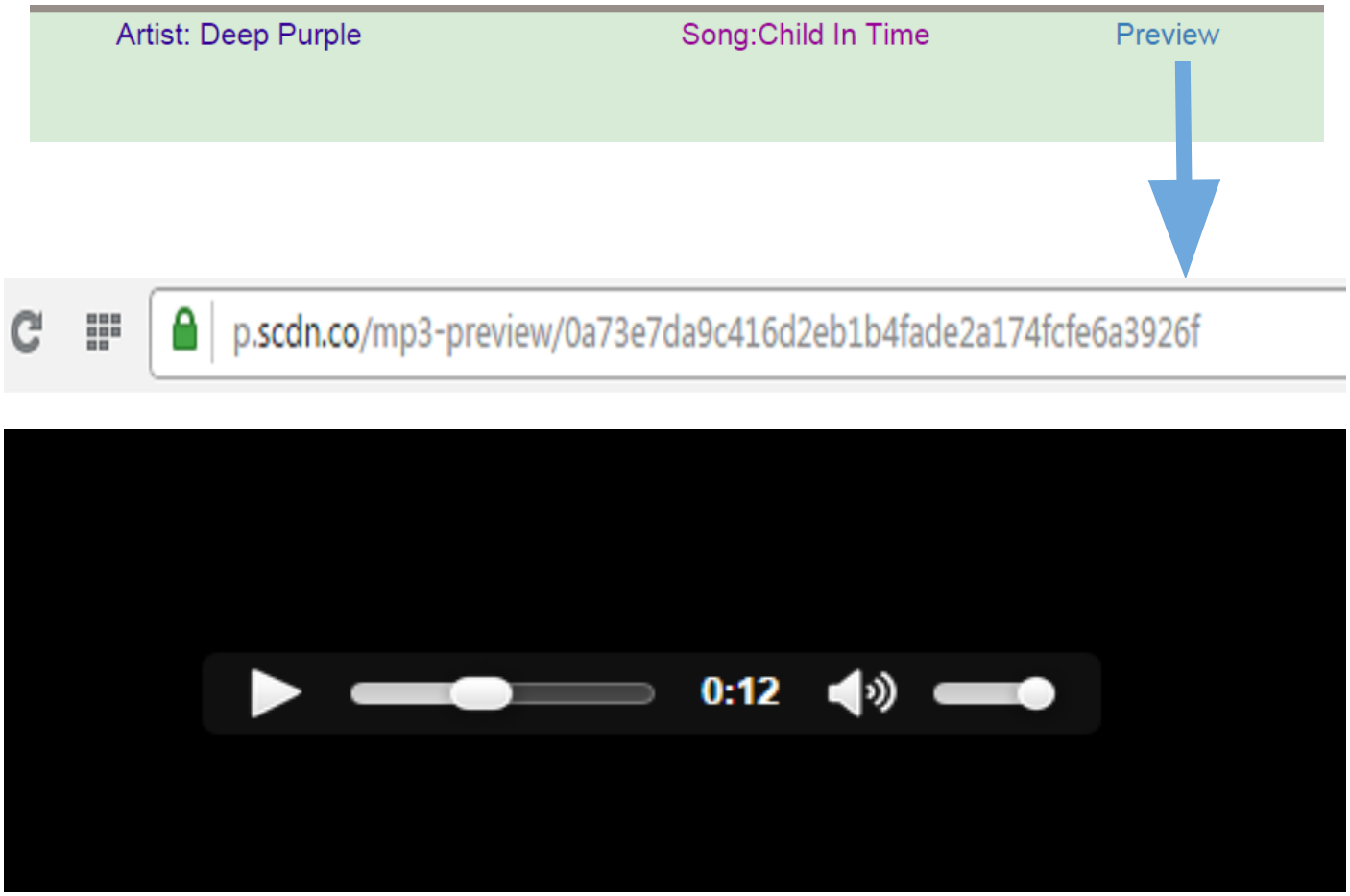
  Facade

Project: songFinder

songFinder

Song Search Engine using Spotify web services

songFinder - Song Preview



songFinder - Single Result

On single result the song plays automatically

songFinder - Drawbacks

- **No caching:** everytime long list of songs is being fetched and rendered

- **Tight coupling:** parsing and rendering the ws response in one place

Proxy Design Pattern

Proxy Pattern - Definition

- ★ Provides a **surrogate** for another object
- ★ Uses an extra level of **indirection**
- ★ Adds a **wrapper** to protect the real component from undue complexity

Proxy Pattern - Applicability

Cache Proxies improve performance of underlying members

Virtual Proxies delay initialization of expensive objects

Remote Proxies represent locally a remote object

Protection Proxies control access to a sensitive object

Smart Proxies interpose additional actions when an object is accessed

songFinder - Refactoring using Proxy

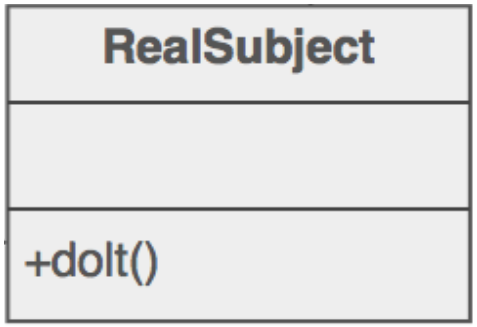
Before

```
$('form').on('submit', function(){
  event.preventDefault();
  $("#song-container").empty();
  //$("form").append($("<img src='loading.gif' alt='loading' >") );

  //console.log("I was clicked");
  //console.log("songInputVal-" +songInput.val());
   $.get('https://api.spotify.com/v1/search?q='+ $songInput.val() + '&type=track', function(data) {
     console.log(data.tracks.total);
     if(data.tracks.total){
      //console.log(getSongArray(data) );
      //console.log(createSongsObj(getSongArray(data), Song));
      display(createSongsObj(getSongArray(data), Song) );
    }else{
      $('#song-container').append("<div class='errorInput text-center'>Err0r enter a real song</div>");
      //$('#song-container').addClass("error");
    }
    ....
    ....
```

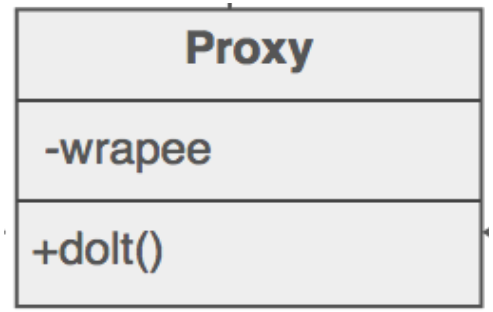
songFinder - Refactoring using Proxy

```
33  //calls a web service to search for a song
34  function SongWS() {
      this.getSong = function(songInput) {
36      var url = 'https://api.spotify.com/v1/search?q='+ songInput + '&type=track
37      var response = undefined;
38      //TODO: refactor - sync ajax call is deprecated
39      $.ajax({
40        url: url,
41        success: function(data) {
42          response = data;
43        },
44        async:false
45      });
46      return response;
47    };
48  }
```



songFinder - Refactoring using Proxy (cont)

```
51  //caches frequently requested songs. If a song is not already cached
52  function SongProxy() {
53    var songws = new SongWS();
54    return {
        getSong: function(songInput) {
56          if (!songcache[songInput]) //cache miss -> add to cache
57              songcache[songInput] = songws.getSong(songInput);
58          return songcache[songInput];
59      },
60      getCount: function() {
61          var count = 0;
62          for (var song in songcache) { count++; }
63          return songcache.count;
64      }
65    };
66  };
67
```



songFinder - Refactoring using Proxy (cont)

- ★ Provides an **interface similar** to the real object
- ★ **Reference** that lets the proxy access the real object
- ★ Handles requests and **forwards** these to the real object

```
132 $('form').on('submit', function(){
133     event.preventDefault();
134     $("#song-container").empty();
135     //create a songProxy instance to search song in cache and redirect if necessary
136     var songproxy = new SongProxy();
137     // execute songproxy request and store result
138     var response = songproxy.getSong($songInput.val());
139     //display response
140     display(response);
```

Proxy Pattern - Trade-off

Less efficiency due to indirection

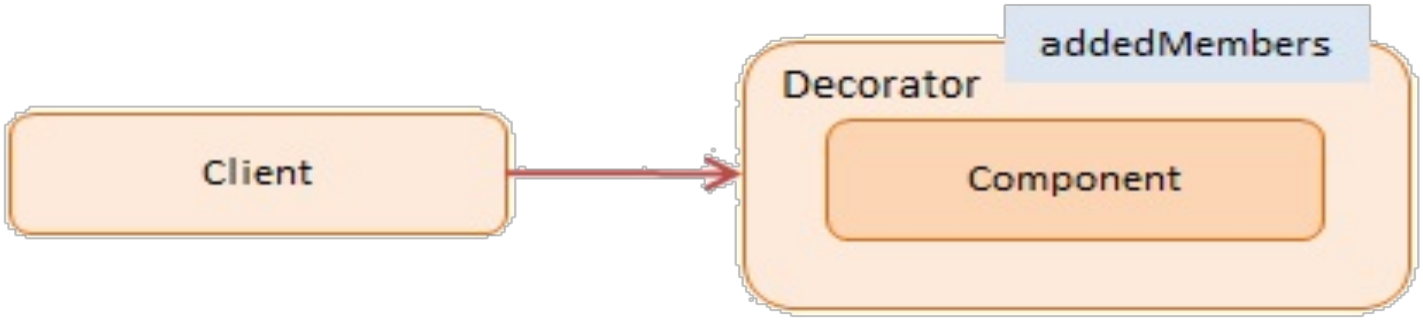
Complex implementation

Decorator Design Pattern

Decorator Pattern - Definition

- ★ Attaches additional functionality dynamically and remove functionality at a later time
- ★ An alternative to subclassing for extending functionality
- ★ Multiple decorators can add functionality to original object
- ★ A.k.a. **Wrapper**

Decorator Pattern - Structure



songFinder - Refactoring using Decorator I

Simple Song Object

```
7   function Song(title,artist, img, playUrl){
8     this.title = title;
9     this.artist = artist;
10    this.img = img;
11    this.play = playUrl || "#";
12
13    this.render = function(templateSource, templateLocation){
14      var $songTemplate = _.template( $(templateSource).html() );
15      var $songLocation = $(templateLocation);
16      $songLocation.append($songTemplate(this) );
17    }
18  }
19
```

songFinder - Refactoring using Decorator I

Decorated Song Object - Playable Song

```
20  function PlayableSong(song){
21    this.title = song.title;
22    this.artist = song.artist;
23    this.img = song.img;
24    this.play = song.play || "#";
25    var playUrl = this.play;
26
27    this.render = function(templateSource, templateLocation){
28      var $songTemplate = _.template( $(templateSource).html() );
29      var $songLocation = $(templateLocation);
30      $songLocation.append($songTemplate(this) );
31      //show audio controls and play song
32      var audio = document.getElementById("songAudio");
33      audio.setAttribute("controls","controls");
34      document.getElementById("spotifySong").src = playUrl;
35      audio.play(playUrl);
36    }
37  }
38
```

songFinder - Single Result

On single result the song plays automatically

songFinder - Refactoring using Decorator II

```
function SongView(data) {
    this.data = data;
    this.decorator;
    this.render = function(){
      if(this.decorator) {
        //if decorator used render with decorated view
        decorators[this.decorator].render(this.data);
        return;
      }
      //since its a single song instantiate playableSong
      var playableSong = new PlayableSong(getSongObj(data.tracks.items[0]))
      playableSong.render("#song-template", "#song-container");
    }

    this.decorate = function(decorator){
      this.decorator = decorator;
    }
}
```

songFinder - Refactoring using Decorator II

```
else if (response.tracks.items.length > 1) { // If multiple potential songs found
  var songView = new SongView(response);
  songView.decorate('songsView');
  songView.render();
}
```

```
// view decorators
var decorators = {};

//create decorator for rendering multiple songs view
decorators.songsView = {
    render: function(data) { //custom implementation of render
        _.each(data.tracks.items, function(songObj, index){
          getSongObj(songObj).render("#song-template", "#song-container");
      })
    }
};
```

songFinder - Refactoring using Decorator II

We achieved to:

- ★ change efficiently the Song object behavior based on the response format at runtime
- ★ provide a **clean** and **object-oriented** way to handle and render different server responses

Decorator Pattern - Trade-off

Overuse of decorators can result in many small objects that means…

maintainance headache

Outline

- ❑ Structural Design Patterns
- ❑ Applied Structural Patterns
- ❑ Project 1: songFinder
- ❑ Proxy
- ❑ Decorator
- ❑ Facade
- ❑ Composite
- ❑ Project 2: drawr-bootstrap
- ❑ Q&A

Façade Design Pattern

Façade Pattern - Definition

- ★ Convenient higher-level interface to a larger body of code
- ★ Hides actual complexity
- ★ Simplified presentation of API - Improves usability
- ★ Decouples class from code that utilizes it

Example

Withdraw 100 €

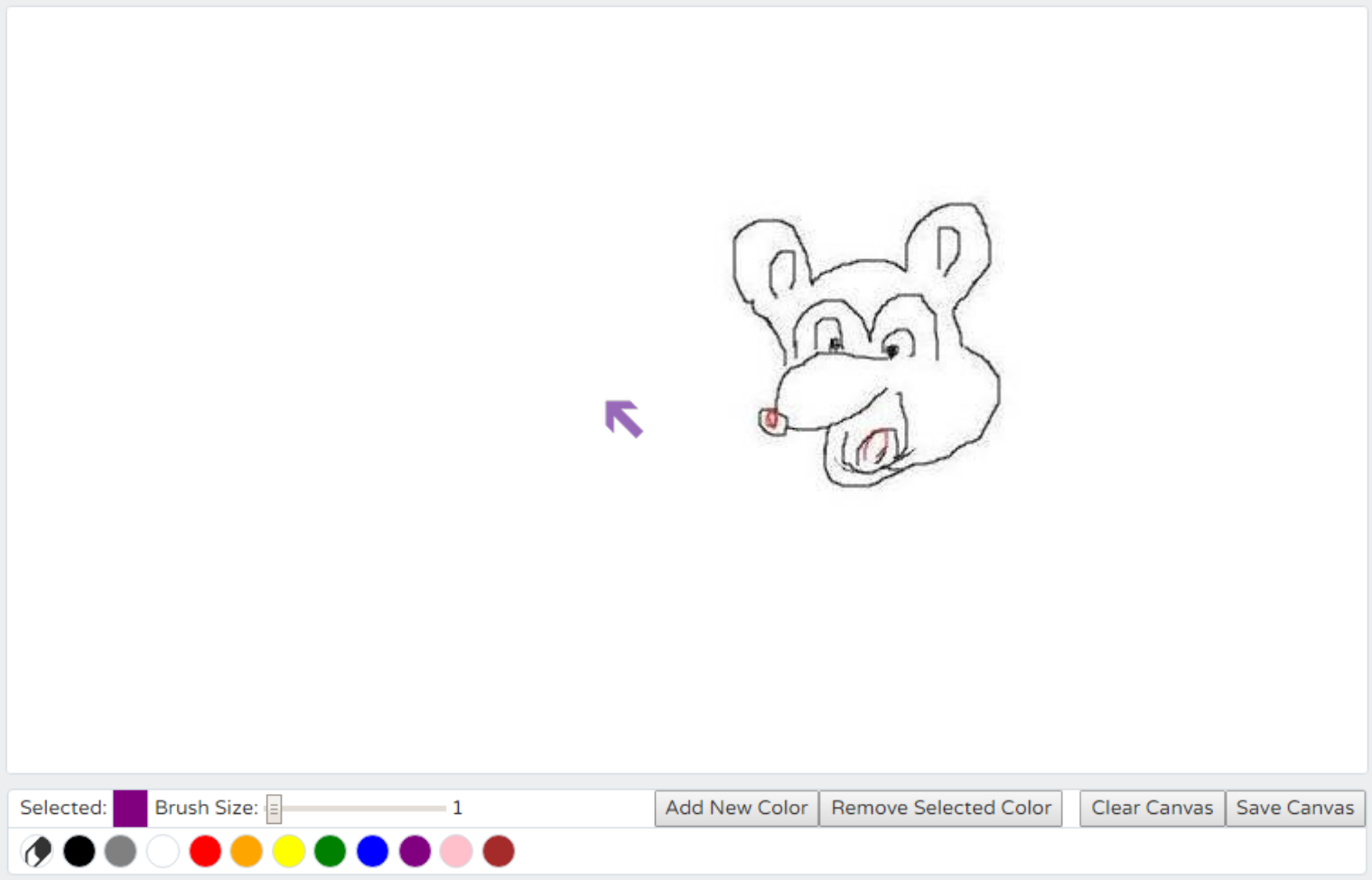
Example

Withdraw 100 €

- Check account validity
- Verify PIN
- Check funds availability
- Make further updates

Project : drawr-bootstrap[1]

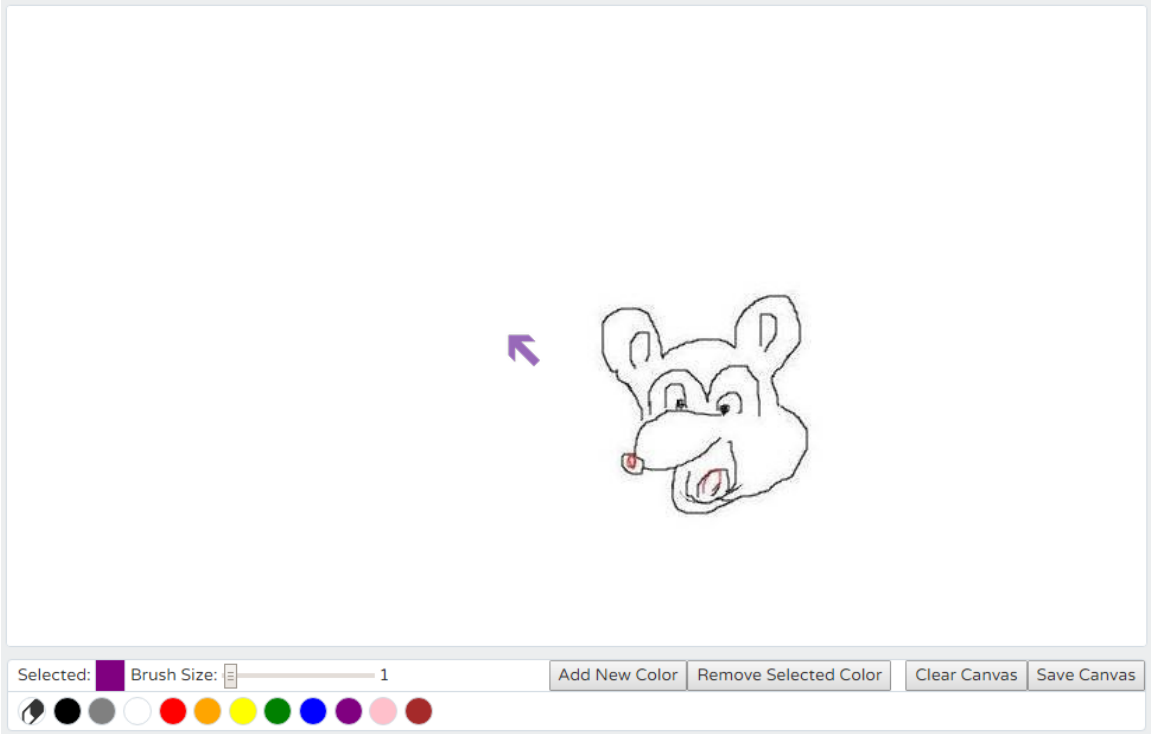
drawr-bootstrap



First Screen

drawr-bootstrap: features

- Customizable drawing tool
- Toolbar
  - Brush/Eraser thickness
  - Color Palette
  - Add new color
  - Remove a color
  - Clear canvas
  - Save canvas



Applying Façade Design Pattern

Several event listeners & handlers

- $('**#palette**').on('**click**', '**li**', **function** () {…}
- $('**#erase**').click(**function** () { …}
- $('**#thickness**').change(**function** () {…}
- $('**#eraserthickness**').change(**function** () {…}
- $('**#save**').click(**function** () {…}
- $('**#clear**').click(**function** () {…}
- $('**#addcolor**').click(**function** () {…}
- $('**#removecolor**').click(**function** () {…}
- $('**#attachcolor**').click(**function** () {…}
- $('**#cancelcolor**').click(**function** () {…}
- $('**.colorslider**').change(**function** () {…}

Original Version

index.html × app.js ×

```javascript
$(document).ready(function () {
  var color = $('.selected').css('background-color');
  var paintSurface = $('#paintsurface');
  var ctx = paintSurface[0].getContext('2d');
  var lastEvent;
  var canvasClicked = false;
  var thickness = $('#thickness').val();

  $('#palette').on('click', 'li', function () { // when color palette items selected:
    color = $(this).css('background-color'); // set brush color to color selected
    $(this).siblings().removeClass('selected');
    $(this).addClass('selected'); // make clicked color 'selected'
    changeCursor(color); // change cursor to new color
    thickness = $('#thickness').val(); // update thickness
    $('#thickcounter').text(thickness); // change thickness counter to match new thickness
    $('#thickness').removeAttr('disabled');
    $('#eraserthickness').attr('disabled','disabled');
    $('.brushcontrol').show();
    $('.erasercontrol').hide(); // show/enable brush thickness slider, hide/disable eraser thickness slider
    $('#selectedtool').css('background', color); // update selected tool with new color
  });

  $('#thickness').change(function () { // change thickness when size slider changed and update counter
    thickness = $('#thickness').val();
    $('#thickcounter').text(thickness);
  });

  $('#eraserthickness').change(function () { // change eraser thickness when eraser slider changed and update counter
    thickness = $('#eraserthickness').val();
    $('#thickcounter').text(thickness);
  });

  $('#save').click(function () { // fetch canvas url and open in new tab to save
    var dataURL = paintSurface[0].toDataURL('image/png');
```

app.js

Limitations

- All event listeners/handlers defined in a single function
- Confusing! No abstraction
- Browsers' compatibility not checked

index.html × app.js ×

```
$(document).ready(function () {
  var color = $('.selected').css('background-color');
  var paintSurface = $('#paintsurface');
  var ctx = paintSurface[0].getContext('2d');
  var lastEvent;
  var canvasClicked = false;
  var thickness = $('#thickness').val();

  $('#palette').on('click', 'li', function () { // when color palette items selected:
    color = $(this).css('background-color'); // set brush color to color selected
    $(this).siblings().removeClass('selected');
    $(this).addClass('selected'); // make clicked color 'selected'
    changeCursor(color); // change cursor to new color
    thickness = $('#thickness').val(); // update thickness
    $('#thickcounter').text(thickness); // change thickness counter to match new thickness
    $('#thickness').removeAttr('disabled');
    $('#eraserthickness').attr('disabled','disabled');
    $('.brushcontrol').show();
    $('.erasercontrol').hide(); // show/enable brush thickness slider, hide/disable eraser thickness slider
    $('#selectedtool').css('background', color); // update selected tool with new color
  });

  $('#thickness').change(function () { // change thickness when size slider changed and update counter
    thickness = $('#thickness').val();
    $('#thickcounter').text(thickness);
  });

  $('#eraserthickness').change(function () { // change eraser thickness when eraser slider changed and update counter
    thickness = $('#eraserthickness').val();
    $('#thickcounter').text(thickness);
  });

  $('#save').click(function () { // fetch canvas url and open in new tab to save
    var dataURL = paintSurface[0].toDataURL('image/png');
```

app.js

a simple facade that masks the various browser-specific methods

```
89
90  function addEvent( element, event, callback ) {
91      if( window.addEventListener ) {
92          element.addEventListener( event, callback, false );
93      } else if( document.attachEvent ) {
94          element.attachEvent( 'on' + event, callback );
95      } else {
96          element[ 'on' + event ] = callback;
97      }
98  }
```

app.js

Browser-specific methods

```
89
90  function addEvent( element, event, callback ) {
91      if( window.addEventListener ) {
92          element.addEventListener( event, callback, false );
93      } else if( document.attachEvent ) {
94          element.attachEvent( 'on' + event, callback );
95      } else {
96          element[ 'on' + event ] = callback;
97      }
98  }
```

1.0	9.0	1.0	1.0	7.0

```
89
90 function addEvent( element, event, callback ) {
91     if( window.addEventListener ) {
92         element.addEventListener( event, callback, false );
93     } else if( document.attachEvent ) {
94         element.attachEvent( 'on' + event, callback );
95     } else {
96         element[ 'on' + event ] = callback;
97     }
98 }
```

app.js

```
1  var color = $('.selected').css('background-color');
2  var paintSurface = $('#paintsurface');
3  var ctx = paintSurface[0].getContext('2d');
4  var lastEvent;
5  var canvasClicked = false;
6  var thickness = $('#thickness').val();
7
8  addEvent($('#addcolor'), 'click', function() {
9      addColorClicked();
10 });
11 addEvent($('#removecolor'), 'click', function() {
12     removeColorClicked();
13 });
14 addEvent($('#attachcolor'), 'click', function() {
15     attachColorClicked();
16 });
17 addEvent($('#cancelcolor'), 'click', function() {
18     cancelColorClicked();
19 });
20 addEvent($('#clear'), 'click', function() {
21     clearClicked();
22 });
23 addEvent($('#save'), 'click', function() {
24     saveClicked();
25 });
```

Achieving abstraction

```
61  function removeColorClicked() {
62      $('li.selected').remove();
63      $('#palette li:last-child').click();
64  }
65
66  function attachColorClicked() {
67      var newColor = $('<li></li>');
68      newColor.css('background-color', $('#colorpicked').css('background-color'));
69      $('#palette').append(newColor);
70      addEvent(newColor[0], 'click', function() {
71          var returnValues = paletteClicked(this);
72          color = returnValues[0];
73          thickness = returnValues[1];
74      });
75      $('#colorpicker').hide();
76  }
77
78  function cancelColorClicked() {
79      $('#colorpicker').hide();
80  }
81
82  function clearClicked() {
83      var paintSurface = $('#paintsurface');
84      var ctx = paintSurface[0].getContext('2d');
85      ctx.fillStyle = 'rgb(255,255,255)';
86      ctx.lineWidth = 0;
87      ctx.clearRect(0,0,960,540); // erase canvas
88      ctx.rect(0,0,960,540);
89      ctx.stroke();
90      ctx.fill(); // make background white instead of tansparent
91  }
92
93  function saveClicked() {
```

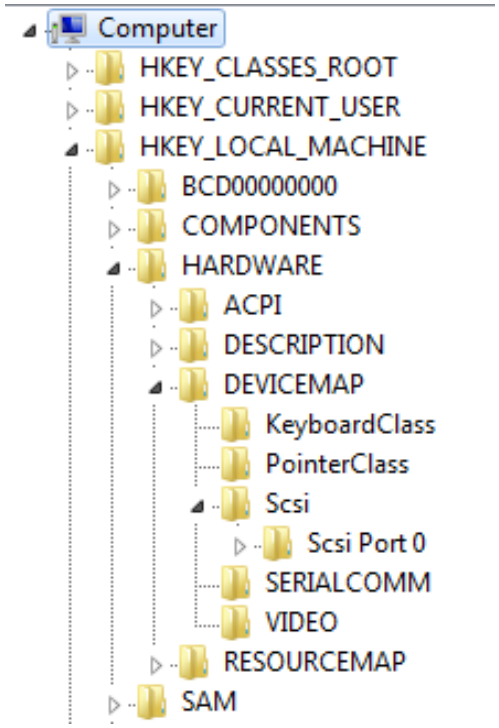
eventHandler.js

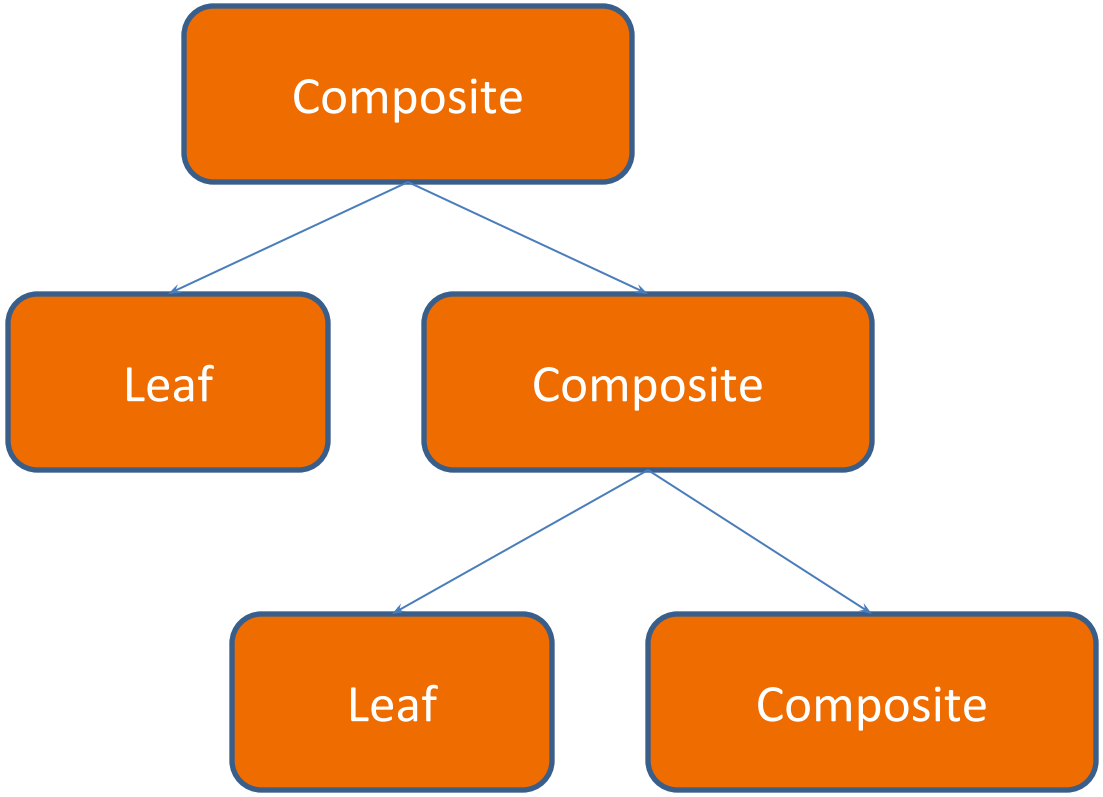
Composite Design Pattern

Composite Pattern - Introduction

- ★ Group of objects should be treated in the same way as a single instance of an object.
- ★ Same behavior applied to an object or a group of objects
- ★ Code resuability

Example


Computer
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
BCD00000000
COMPONENTS
HARDWARE
ACPI
DESCRIPTION
DEVICEMAP
KeyboardClass
PointerClass
Scsi
Scsi Port 0
SERIALCOMM
VIDEO
RESOURCEMAP
SAM


Composite
Leaf
Composite
Leaf
Composite

Applying Composite Design Pattern

Original version

```
42  $('#palette').on('click', 'li', function () { // when color palette items selected:
43    color = $(this).css('background-color'); // set brush color to color selected
44    $(this).siblings().removeClass('selected');
45    $(this).addClass('selected'); // make clicked color 'selected'
46    changeCursor(color); // change cursor to new color
47    thickness = $('#thickness').val(); // update thickness
48    $('#thickcounter').text(thickness); // change thickness counter to match new thickness
49    $('#thickness').removeAttr('disabled');
50    $('#eraserthickness').attr('disabled','disabled');
51    $('.brushcontrol').show();
52    $('.erasercontrol').hide(); // show/enable brush thickness slider, hide/disable eraser thickness slider
53    $('#selectedtool').css('background', color); // update selected tool with new color
54  });
55
56  $('#thickness').change(function () { // change thickness when size slider changed and update counter
57    thickness = $('#thickness').val();
58    $('#thickcounter').text(thickness);
59  });
60
61  $('#eraserthickness').change(function () { // change eraser thickness when eraser slider changed and update counter
62    thickness = $('#eraserthickness').val();
63    $('#thickcounter').text(thickness);
64  });
65
66  $('#erase').click(function () { // when eraser selected:
67    color = 'white'; // set eraser to white
68    thickness = $('#eraserthickness').val(); // update thickness
69    $('#thickcounter').text(thickness); // change thickness counter to match new thickness
70    $(this).removeClass('selected'); // don't select
```

Component with collection of list elements

Single element component

Refactored version

```
69  addEvent($('#palette'), 'click', function(){
70      var returnValues = paletteClicked(this);
71      color = returnValues[0];
72      thickness = returnValues[1];
73  });
74  addEvent($('#paintsurface'), 'mousedown', function(e){
75      var returnValues = mouseDownOnCanvas(e, canvasClicked, lastEvent);
76      lastEvent = returnValues[0];
77      canvasClicked = returnValues[1];
78  });
79  addEvent($('#erase'), 'click', function() {
80      var returnValues = eraseClicked();
81      color = returnValues[0];
82      thickness = returnValues[1];
83  });
84  addEvent($('.colorslider'), 'change', function() {
85      colorSliderChanged(this);
86  });
87
```

Component with collection of list elements

Single element component

Refactored version

```
90  function addEvent( element, event, callback ) {
91      console.log("element : " + element);
92      if(element.nodeName == "UL") {
93          console.log("List ...");
94          for(var i = 0; i < element.children.length; i++) {
95              console.log("count : " + i);
96              if(element.children[i].nodeName == 'LI')
97              addEvent(element.children[i], event, callback);
98          }
99      } else if(element.length > 0) {
100         console.log("multiple elements ..");
101         for(var i = 0; i < element.length; i++) {
102             addEvent(element[i], event, callback);
103         }
104     }
105     else {
106         console.log("element class : " + element.className);
107         if( window.addEventListener ) {
108             element.addEventListener( event, callback, false );
109         } else if( document.attachEvent ) {
110             element.attachEvent( 'on' + event, callback );
111         } else {
112             element[ 'on' + event ] = callback;
113         }
114     }
115 }
```

Recursion for Collection components

Composite Pattern - Trade-off

- Once tree structure is defined, the composite design makes the tree overly general.
- In specific cases, it is difficult to restrict the components of the tree to only *particular types*.

References

[1] Stefanov, S., JavaScript Patterns, O'Reilly Media, 2010

[2] Sourcemaking, Design Patterns Explained Simple,

https://sourcemaking.com/design_patterns/structural_patterns

[3] dofactory, Javascript Design Patterns,

http://www.dofactory.com/javascript/design-patterns

[4] gofpatterns, Gang of Four Patterns, http://www.gofpatterns.com

[5] github, flamingveggies/drawr,

https://github.com/flamingveggies/drawr

[6] github, goodbedford/songFinder,

https://github.com/goodbedford/songFinder

Q&A

Thank you

Questions?

Chrysa Papadaki // chr.papadaki@tum.de

Nishant Gupta // nishant.gupta@tum.de