Client-side Frameworks

Farooq Ahmed, FAST-NU, Lahore

JQuery

- Cross-browser Javascript library for DOM manipulation
- Combines CSS selectors with Javascript for expressive selection
- Built around the philosophy
  - Find elements
  - Perform actions
- Maintains a nice separation between HTML and Javascript
- Plugin-based architecture for extensions
- Many cross-browser reusable widgets in JQuery UI

Basic Javascript Example

```
<html>
        <head>
                <script type="text/javascript" >
                        function toggleParagraphs(){
                                var elements = document.getElementsByTagName(“p”);
                                for(i=0; i < elements.length; i++){
                                        if(elements[i].style.display != “none”){
                                                elements[i].style.display = none;
                                        }
                                        else{
                                                elements[i].style.display = “block”;
                                        }
                                }
                        }
                </script>
        </head>

        <body>
                <p> paragraph-1 </p>
                <p> paragraph-2 </p>
                <p> paragraph-3 </p>
                <button id="button" onclick=”toggleParagraphs()”>Toggle</button>
        </body>
</html>
```

JQuery Example

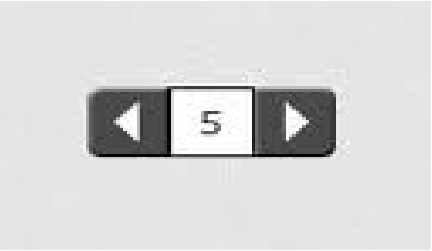
```
<html>
	<head>
		<script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>

		<script type="text/javascript" >
			$( function (){
					$("#button").click(

						function(){
							$("p").toggle();
						}
					)
				}
			);
		</script>
	</head>

	<body>
		<p> paragraph-1 </p>
		<p> paragraph-2 </p>
		<p> paragraph-3 </p>
		<button id="button">Toggle</button>
	</body>
</html>
```

Example



Write code for a Javascript-based Spin control as shown in the figure above.

Spin control shall have following elements:

- A text field to display a number. Programmer shall be able to specify which number to show when the counter is first displayed

- Increment button that will advance the counter by the step size. For example if the current value is 5 and step size is 5, pressing Increment button should show 10 in the text field. Programmer shall be able to specify step size.

- Decrement button that will reduce the counter by the step size. For example if the current value is 5 and step size is 5, pressing Decrement button should show 0 in the text field.

Please note the following constraints:

- Programmer shall be able to specify which number to show when the counter is first displayed

- Programmer shall be able to specify step size.

- Programmer shall be able to get the current value being displayed in control through a function defined in control

Basic Javascript Spin Control

```
var SpinPool = new Array();

function Spin(sn,ss,id){
	this.value = sn;
	this.step = ss;
	this.id = id;

	SpinPool[id] = this;
}

Spin.prototype.getFieldId = function(){
	return this.id + "_textfield";
}

Spin.prototype.increment = function(){
	this.value += this.step;
	document.getElementById(this.getFieldId()).value = this.value;
}

Spin.prototype.decrement = function(){
	this.value -= this.step;
	document.getElementById(this.getFieldId()).value = this.value;
}

Spin.prototype.get = function(){
	return this.value;
}
```

Basic Javascript Spin Control

```
Spin.prototype.display = function(id) {
        var html = "<input type='button' value='<' onclick='spin_previous(\"" + this.id + "\")'></input>";
        html += "<input type='text' value='" + this.value + "' id='" + this.getFieldId() + "'></input>";
        html += "<input type='button' value='>' onclick='spin_next(\"" + this.id + "\")'></input>";
        document.getElementById(id).innerHTML = html;

}


function spin_previous(id){
        SpinPool[id].decrement();
}

function spin_next(id){
        SpinPool[id].increment();
}
```

Using Javascript Spin Control

```
<html>
 <head>
        <script type="text/javascript" src="./spin.js"></script>
        <script type="text/javascript">

                function load(){
                        var obj = new Spin(5,5,"spin1");
                        obj.display('spin_area');

                        var obj = new Spin(2,2,"spin2");
                        obj.display('spin2_area');
                }

        </script>
 </head>
 <body onload="load()">
   <div id='spin_area'></div>
   <div id='spin2_area'></div>
 </body>
</html>
```

Creating JQuery Plugin

```
Spin.prototype.getHtml = function() {
	var html = "<input type='button' value='<' onclick='spin_previous(\"" + this.id + "\")'></input>";
	html += "<input type='text' value='" + this.value + "' id='" + this.getFieldId() + "'></input>";
	html += "<input type='button' value='>' onclick='spin_next(\"" + this.id + "\")'></input>";
	return html;
}

// ------------------------

$.fn.spinner = function(iv,ss){

	return this.each(function(){
		var obj = new Spin(iv,ss,this.id);
		$(this).html(obj.getHtml());
	});
}
```

Using JQuery Plugin

```
<html>
 <head>
        <script src="../jquery-1.11.3.min.js"></script>
        <script src="spin.js"></script>
        <script type="text/javascript">

                $( function(){
                        $("#spin1").spinner(5,5);
                        $("#spin2").spinner(2,2);
                });

        </script>
 </head>
 <body>
   <div id='spin1'></div>
   <div id='spin2'></div>
 </body>
</html>
```

Angular JS

- Javascript framework for client side MVC applications
- Intends to decouple DOM manipulation from application logic
- Supports data binding
  - Useful for single page applications
- Operates by extending HTML vocabulary
  - Create and use new directives

Basic example

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

Basic example

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p>{{name}}</p>
</div>

</body>
</html>
```

Creating AngularJS Directive

```
var app = angular.module("spinApp",[]);

app.directive("spinWidget",function(){

	return {
		scope : {},
		link : function(scope,element,attrs){
			alert(attrs.sn + "," + attrs.sn + "," + attrs.id );
			scope.spinner = new Spin(Number(attrs.sn),Number(attrs.ss),attrs.id);
			element.append(scope.spinner.getHtml());
		}
	};
});
```

Using AngularJS Directive

```
<html>
 <head>
    <script src="../angular.min.js"></script>
          <script src="spin.js"></script>

 </head>
 <body  ng-app="spinApp">
   <spin-widget sn="5" ss="5" id="spin1" ></spin-widget>

   <spin-widget sn="1" ss="1" id="spin2" ></spin-widget>
 </body>
</html>
```

React JS

- Component-based development
  - Stateless
  - Statefull
- Virtual DOM
  - Detection of changes in component state
  - Efficient updates to actual DOM
- Declarative programming support
  - JSX (Javascript XML)

Creating React Component

```
class Spin extends React.Component{

        constructor(props){
                super(props);
                this.state = {value: this.props.sn};
        }


    render(){

                var html = React.createElement("span",null,this.getHtml());

            return html;
    }

    …

}
```

```
class Spin extends React.Component{

    constructor(props){
                        super(props);
                        this.state = {value: this.props.sn};
                        this.increment = this.increment.bind(this);
                        this.decrement = this.decrement.bind(this);
    }

    render(){
            var html = React.createElement("span",null,this.getHtml());
            return html;
    }

    getHtml(){

            var elems = new Array();
            elems[0] = React.createElement("input",{type:'button',value:'<', onClick: this.decrement});
            elems[1] = React.createElement("input",{type: 'text', value:this.state.value});
            elems[2] = React.createElement("input",{type:'button',value:'>', onClick: this.increment});

            return elems;
    }

    decrement(){
            this.setState({value:(this.state.value-this.props.ss)});
    }

             increment(){
            this.setState({value:(this.state.value+this.props.ss)});
    }
}
```

Using React Component

```
<html>
 <head>
    <script src="https://unpkg.com/react@16/umd/react.development.js" ></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
    <script src="spin.js"></script>
    <script type="text/javascript">

                function load(){
                        ReactDOM.render(React.createElement(Spin,{sn:1,ss:1,id:'spin1'}),
                                document.querySelector('#divSpin1'));

                        ReactDOM.render(React.createElement(Spin,{sn:5,ss:5,id:'spin2'}),
                                document.querySelector("#divSpin2"));
                }
        </script>
 </head>
 <body>
   <div id='divSpin1'></div>
   <div id='divSpin2'></div>
 </body>
</html>
```