WED3 FS18 Cheat Sheet – Marc Etter

SPA

Pro: No Page Reloads, Granular actions are fast, development of complicated features is easier, almost native experience

Con: Broken “back” and “open in new tab” buttons, broken “refresh”, slow?

Vue.js

Lifecycle: Created -> Mounted -> Updated -> Destroyed

Templates

```
<div id=”app”>
  <h2 class=”hello-title”>Hello {{name}}!</h2>
</div>
<script>
  var vm = new Vue({ el: ‘#app’, data: { name: ‘World’ }, created:
    function () {
      fetch(‘/spa/’+window.location.hash.substring(1))
        .then(response => response.json())
        .then(body => { this.name = body.value; });
    }});
</script>
```

Components

```
// register
Vue.component(‘my-component’, {
  Template: ‘<div>A custom component!</div>’
});
// create a root instance
new Vue({ el: ‘#example’ });
// html file
<div id=”example”>
  <my-component></my-component>
</div>
// will produce
<div id=”example”>
  <div>A custom component!</div>
</div>
```

React.js

React-Elements must be upper-case.

No JavaScript keywords inside JSX (use && or ?:, and className)

Reconciliation: render virtual DOM, diff, update changed parts of real DOM with updated state.

Lifecycle

Mounting: constructor(props), render(), componentDidMount()

Updating: render()

Unmounting: componentWillUnmount()

Skeleton

```
class Login extends React.Component {
  state = { username: ‘Benutzername’ }
  handleNameChange = (event) => {
    this.setState({username: event.target.value});
  }
  handleSubmit = (event) => {
    event.preventDefault();
    this.props.authenticate(this.state.username);
  }
  render() {
    return (
      <form>
        <input type=”text” value={this.state.username}
            onChange={this.handleNameChange} />
        <button type=”submit” onClick={this.handleSubmit}>Login</button>
      </form>);
  }
}
```

Angular

Skeleton

/welcome: entry-page

/dashboard: lazy-loaded, auth-protected

app.module.ts

```
@NgModule({ declarations: [ AppComponent ],
  imports: [ …, CoreModule.forRoot(), AuthModule.forRoot(),
    WelcomeModule.forRoot(), AppRoutingModule ],
  providers: [], bootstrap: [ AppComponent ] })
export class AppModule { }
```

app.component.ts

```
@Component({ selector: ‘app-root’, templateUrl: ‘./app.component.html’,
  styleUrls: [ ‘app.component.scss’ ] })
export class AppComponent { }
```

app-routing.module.ts

```
const routes: Routes = [{
  path: ‘dashboard’, canLoad: [ AuthGuard ],
  loadChildren: ‘app/dashboard/dashboard.module#DashboardModule’ },
  { path: ‘’, redirectTo: ‘/welcome’, pathMach: ‘full’ }];
@NgModule({ imports: [ RouterModule.forRoot(routes) ],
  exports: [ RouterModule ] })
export class AppRoutingModule { }
```

welcome.module.ts

```
@NgModule({ declarations: [ WelcomeComponent ],
  imports: [ WelcomeRoutingModule, SharedModule, … ],
  exports: [ WelcomeComponent ], providers: [ ] })
export class WelcomeModule {
  static forRoot(config?:{}) : ModuleWithProviders {
    return { ngModule: WelcomeModule, providers: [ ] };
  }
}
```

welcome-routing.module.ts

```
const routes: Routes = [{ path: ‘welcome’, component: WelcomeComponent }];
@NgModule({ imports: [ RouterModule.forChild(routes) ],
  exports: [ RouterModule ] })
export class WelcomeRoutingModule { }
```

dashboard.module.ts

```
@NgModule({ declarations: […], imports: [ DashboardRoutingModule, … ],
  exports: [ ], providers: […] })
export class DashboardModule {
  static forRoot(config?:{}) : ModuleWithProviders {
    return …;
  }
}
```

dashboard-routing.module.ts

```
const routes: Routes = [{
  path: ‘dashboard’, component: DashboardComponent, children = [
    { path: ‘’, component: LandingComponent }] };
@NgModule({ imports: [ RouterModule.forChild(routes) ],
  exports: [ RouterModule ] })
export class DashboardRoutingModule { }
```

dashboard.component.html

```
<wed-frame title=”- Dashboard”>
  <wed-navigation navigation class=”navbar-collapse”>
    <ul class=”navbar-nav mr-auto”>
      <li class=”nav-item”>
        <a class=”nav-link” routerLink=”/welcome”
          routerLinkActive=”active”>Home</a>
      </li>
    </ul>
  </wed-navigation>
  <main class=”container=fluid”>
    <wed-landing></wed-landing>
  </main>
</wed-frame>
```

dashboard.component.ts

```
@Component({ … }) export class DashboardComponent { }
```

landing.component.html

```
<wed-card title=”Your Place – News”>
  <section *ngFor=”let item of favoritePlaces”>
    <header><h3>{{item.title}}</h3></header>
    <div>{{item.description}}</div>
    <footer>{{item.footer}}</footer>
  </section>
</wed-card>
```

landing.component.ts

```
@Component({ selector: ‘wed-landing’, templateUrl:‘landing.component.html’,
  styleUrls: [ ‘landing.component.scss’ ] })
export class LandingComponent implements OnInit {
  public favoritePlaces:PlaceViewModel[];
  ngOnInit() {
    this.getDataFromStorage().subscribe(models => {
      this.favoritePlaces = models; });
  }
}
interface PlaceViewModel { title: string, … }
```

Advanced

ViewEncapsulation: Native, Emulated, None

Transclusion

Child-Content is rendered inside `<ng-content>` (select-Attribute can be used to render selectively, e.g. `select=”[wed-title]”`).

Services / EventEmitters

Create emitter instance: `new EventEmitter<SampleModel[]>();`

Emit an event: `myEmitter.emit(models);`

Subscribe to emitter: `sub = myEmitter.subscribe((data) => { … });`

Unsubscribe from emitter: `sub.unsubscribe();`

JWT

```
const token = encodeBase64Url(header) + ‘.’ + encodeBase64Url(payload) +
  ‘.’ + encodeBase64Url(signature);
```

ASP.NET (MVC)

Middlewares

```
// Register new Middleware (Startup.cs)
app.Use(async (context, next) => { …; await next.Invoke(); …; });
// Create branch for given request-path
app.Map(“/logging”, builder => {
  builder.Run(async (context) => {
    await context.Response.WriteAsync(“Hello World!”);
  });
});
// Terminate request, skip remaining Middlewares
app.Run(async (context) => {
  await context.Response.WriteAsync(“Hello World!”);
});
```

Create custom Middleware:

```
public class CustomMiddleware {
  private readonly RequestDelegate _next;
  public CustomMiddleware(RequestDelegate next) { _next = next; }
  public async Task Invoke(HttpContext context) {
    await _next.Invoke(context); // add custom code
  }
}
// Register our Middleware (Startup.cs)
app.UseMiddleware<CustomMiddleware>();
```

Dependency Injection

Captive Dependency: Incorrectly configured Lifetime

Transient: Created every time requested

Scoped: Created once per web-request

Singleton: Created once

-> Components may only inject deps. with equal or longer lifetime.

Skeleton

Startup.cs

```
public void Configure(IApplicationBuilder app) {
  app.UseMvc(routes => {
    routes.MapRoute(name: “default”, template:
      “{controller=Home}/{action=Index}/{id?}”);
    routes.MapRoute(name: “custom”, template: “custom”, defaults: new {
      controller = “Home”, action = “Index” }); });
}
```

Shared/_Layout.cshtml

```
<html><body>@RenderBody()</body></html>
```

Views/Home/Index.cshtml

```
<form method=”post” action=”/”>
  <label>Echo-Text <input type=”text” name=”echoText” /></label>
  <button type=”submit”>Submit</button>
</form>
```

Views/Home/EchoResult.cshtml

```
<div>Echo-Output: @ViewData[“message”]</div>
```

Controllers/HomeController.cs

```
public class HomeController : Controller {
  public IActionResult Index() {
    return View();
  }
  [HttpPost]
  public IActionResult Index(string echoText) {
    TempData[“message”] = echoText;
    return RedirectAction(“EchoResult”);
  }
  public IActionResult EchoResult() {
    ViewData[“message”] = TempData[“message”] ?? “----“;
    return View();
  }
}
```

IActionResult: View, PartialView, Content, Empty, File, StatusCode (e.g. BadRequest();), Json, Redirect, RedirectToRoute, RedirectToAction

Attributes: [HttpGet], [HttpPost(“/foo”)], [Authorize(Roles=..,Policy=…)], [AllowAnonymous], [Route(“api/[controller]”)], [ValidateAntiForgeryToken]

@model: Reference model in cshtml directly

@model App.Models.Customer (-> @Model.Name)

@inject: Use services incshtml directly

REST-API

Controllers/ValuesController.cs

```
[Route(“api/[controller]”)] // Placed on controller
[HttpGet]
public IEnumerable<Value> Get() { return service.All(); }
[HttpGet(“{id}”)]
public Value Get(int id) { return service.Get(id); }
[HttpPost]
public void Post([FromBody] Value value) { service.Add(value);
  return new CreatedAtActionResult("Get", "Values",
    new { id = service.GetId(value)}, value); }
```

Custom Tag-Helpers

```
public class EmailTagHelper : TagHelper {
  public string MailFor { get; set; }
  public override void Process(TagHelperContext context, TagHelperOutput
      output) {
    output.TagName = “a”; // Replaces <email> with <a>-tag
    output.Attributes.SetAttribute(“href”, “mailto:” + MailFor);
    output.Content.SetContent(MailFor);
  }
}
// Register Helper in Views/_ViewImports.cshtml
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@addTagHelper *, MyTagHelperApp // * = load all, 2nd param = Assembly
```

Policies

```
options.AddPolicy(“Founders”, policy => {
  policy.RequireAuthenticateduser();
  policy.RequireClaim(ClaimTypes.Name, “admin@hsr.ch”, “…”);
});
```

Usage in Razor (cshtml):

```
@inject UserManager<ApplicationUser> manager;
awaitmanager.IsInRoleAsync(user, "Administrator")
```

CSS

```
transition: background 1s ease-in, border 1s ease-in, transform 1s ease-in;
animation: fadein 2s 1 linear; // requires @keyframes fadein {-definition
```

transition: applied when styles change (e.g. hover-effect)

animation: executed immediately (e.g. when adding class or page-load)

Vector Graphics

Rectangle, Circle, Ellipse, **Line, Polyline, Polygon**

Polygon adds a connection from end- to start-point, as opposed to Polyline

```
<svg width="150" height="150" viewBox="0 0 3 3">
  <rect width="5" height="5" x="0" y="0" fill="red"></rect>
  <rect width="3" height="1" x="1" y="2" fill="white"></rect>
  <rect width="1" height="3" x="2" y="1" fill="white"></rect>
</svg>
```

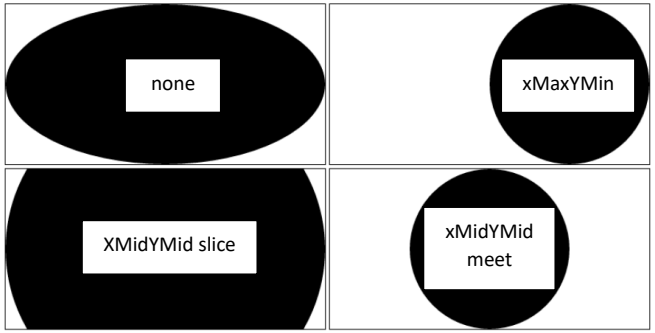
```
<path d="M 100 100 L 300 100 L 200 300 z" fill="orange"
  stroke="black" stroke-width="3" />
```

Upper-case = absolute, Lower-case = relative

M = move, L = draw line, Z = draw line to start, C = Bezier, A = arc

preserveAspectRatio: Alignment: none, min, mid, max (e.g. xMaxYMin)

Reference: meet (scale to smallest dim.) or slice (largest dim.)



Canvas

ctx.translate(…) changes origin for all subsequent operations

ctx.rotate(…) changes rotation for all subsequent operations (Math.PI)

ctx.scale(…) changes scale for all subsequent operations

ctx.moveTo(…) changes start-point of next operation (e.g. for lineTo(…))

ctx.closePath() is optional to end a path (automatic with stroke or fill)

ctx.save() / ctx.restore(): Save current context / restore previous context

ctx.strokeStyle / ctx.fillStyle / ctx.lineWidth: Set draw-style

```
<canvas id="myCanvas" width="50" height="50">
<script>
  var c = document.getElementById("myCanvas");
  var ctx = c.getContext("2d");
  ctx.translate(20, 20);
  ctx.fillRect(0, 0, 5, 5);
  ctx.beginPath();
  ctx.moveTo(10, 0);
  ctx.lineTo(20, 10);
  ctx.lineTo(0, 20);
  ctx.stroke();
</script>
```

Layering: Achieved by creating multiple, overlapping Canvases

Differential-Time (Constant animation regardless of Framerate)

```
function paint(timestamp) {
  ctx.rotate(((timestamp – now) % period / period) * 2 * Math.PI);
  window.requestAnimationFrame(paint);
}
```