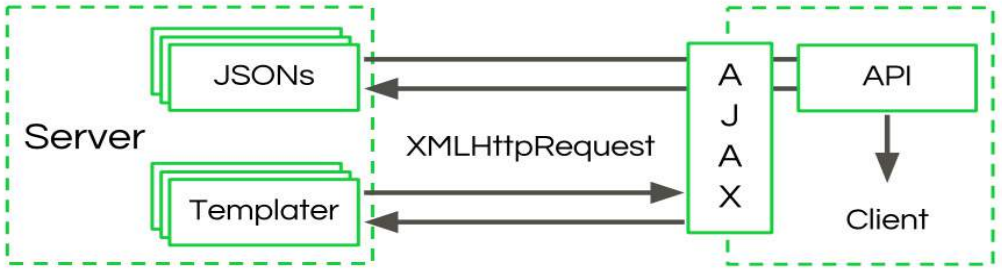
Team 8 – HTML vereinigen / Templating

- 1) Überblick
  // zeige gesamtes Flussdiagramm der Kommunikation Server/Client //



  - GitHub Pages als Server verwenden
  - Daten auf Server: JSON und Templates
  - Client fordert Template mit AJAX von Server an
  - AJAX = Asynchronous Javascript And XML
  - Asynchronität → Templates/Daten können ohne neuen Seitenaufruf aktualisiert werden
  - Template benötigt spezifische Daten
  - Client fordert JSON mit Hilfe der API an
  - API kommuniziert mit Server mit AJAX
  - Verarbeitung durch Programmlogik
  - Template wird mit Daten der Programmlogik befüllt

2) Templating

- Templateengine = handlebars.js
  - Erweiterung von moustache //hier ev Bild von moustache einfügen//
- Nachteile:
  - Komplexe Implementierung
  - Längere Ladezeiten bei vielen Templates
- Vorteile:
  - Keine Logik → nur Tags {{titel}}
    - Kann eingebaute Helfer verwenden, falls nötig → z.B {{#each}}, {{#if}}
  - Einfache Lesbarkeit, Nutzung
  - Tags können mehrere Werte annehmen

3.1) Funktionsweise

- Programmlogik fordert durch AJAX ein Template *//zeige view_templaterender//*
- falls gefordertes Template eine pre_function besitzt → fordere Daten über API
- ansonsten leeres Template
- *//zeige view_callback//* callback teilt view-Funktion mit, wann Template vollständig geladen ist

```
    pre_quizoverview: function (callback) {
        //Random Quiz Vorschlag
        getQuizView(10);

        $(document).on("onQuizView", function (event, data) {
            Quizobject.quizes = data; //copy object

            if (typeof callback == "function") {
                callback(Quizobject);
            }
        });
    },
```

```
    name: "quizoverview",
    sessionobject: {},
    render: function (view, callback) {
        if (this.availableViews.indexOf(view) > -1) {
            var pre_function = this["pre_" + view];
            if (typeof pre_function == "function") {
                pre_function(function (data) {
                    $.get("templates/" + view + ".tpl", function (source) {
                        var template = Handlebars.compile(source);
                        var templatespaceholder = $("#templatespaceholder");
                        templatespaceholder.empty();
                        templatespaceholder.append(template(data));
                        this.name = view;

                        if (typeof callback == "function") {
                            callback();
                        }
                    });
                });
            } else {
                $.get("templates/" + view + ".tpl", function (source) {
                    var template = Handlebars.compile(source);
                    var templatespaceholder = $("#templatespaceholder");
                    templatespaceholder.empty();
                    templatespaceholder.append(template());
                    this.name = view;

                    if (typeof callback == "function") {
                        callback();
                    }
                });
            }
        }
    },
```

3.2) Funktionsweise

//zeige template//

- Template-Datei beinhaltet HTML und handlebar-Tags
- Tags werden mit Daten aus JSON gefüllt *//zeige json//*
- Wird zwischen statischen Header/Footer eingefügt

```
{{#each quizes}}
<li class="js-collapse-section">
    <div class="row qo-quiz-summary js-quiz-summary">

        <!-- Headline -->
        <div class="col-sm-8">
            <a href="#" class="qo-collapse-link js-collapse-link"><i
                        class="qo-arrow ico-light-arrow-right"></i>
                <h2 class="qo-subheadline">{{titel}}</h2></a>
        </div>
        <div class="col-sm-4 _hidden-xs">
            <p class="qo-infos"><strong>{{spielzahl}}</strong> mal gespielt</p>
        </div>
    </div>


    <div class="row qo-quiz-details js-collapse-details _same-height-sm">
        <!-- Quizbild -->
        <div class="col-sm-6 col-md-4">
            <img class="qo-image" src="{{bild}}" alt="{{titel}}">
        </div>
        <!-- End Quizbild -->

        <!-- Quizinfos -->
        <div class="col-sm-6 col-md-8">
            <div class="row">
                <div class="col-sm-12 col-md-10">
                    <p class="qo-text">{{text}}</p>
                </div>
            </div>

            <div class="row qo-quizinfo-box _same-height-xs">
                <div class="col-xs-6 col-sm-4 col-md-6">
                    <p class="qo-time">
                        <time>{{datum}}</time>
                    </p>
                    <p class="qo-author">{{autor}}</p>
                </div>
                <div class="col-xs-6 _visible-xs-block qo-counter">
                    <p><strong>{{spielzahl}}</strong> mal gespielt</p>
                </div>
                <div class="col-sm-8 col-md-6 _hidden-xs">
                    <a data-view=2 href="#" data-quizid="{{this.quizID}}"
                       class="btn qo-button js-change-view">Jetzt spielen!</a>
                </div>
            </div>
            <div class="row qo-quizinfo-box">
                <div class="col-xs-12 _visible-xs-block">
                    <a data-view=2 href="#" data-quizid="{{this.quizID}}"
                       class="btn qo-button js-change-view">Jetzt spielen!</a>
                </div>
            </div>
        </div>
        <!-- End Quizinfos -->
    </div>
</li>
{{/each}}
<!-- End Quizspiel -->
```

```
{
    "quizID":1,
    "titel":"Das Bundesliga Quiz",
    "autor":"AntoninoFederico",
    "datum":"10.11.2016",
    "spielzahl":67,
    "bild":"https://raw.githubusercontent.com/th-koeln/wba1-2(
    "text":"Wie gut kennst du dich mit der Bundesliga aus? Al
},
```

3.3) Funktionsweise

//zeige main_render//

- In main.js können verschiedene Templates über switch-Funktion gerendert werden
- Per default → quizoverview, da Startscreen
- Zudem werden hier restliche Scripte aktiviert (z.B collapse, flickety)

```
89              default:
90                  view.render("quizoverview", function () {
91                      collapse.init();
92                      var flickityConfig = {
93                          // options
94                          cellAlign: 'left',
95                          cellSelector: '.js-carousel-cell',
96                          contain: true,
97                          imagesLoaded: true,
98                          prevNextButtons: false,
99                          setGallerySize: true
100                     };
101                     var $carousel = $('.js-carousel').flickity(flickityConfig);
102                     slideshowNavi.init($carousel);
103                     clicklistener();
104                 });
105                 break;
```