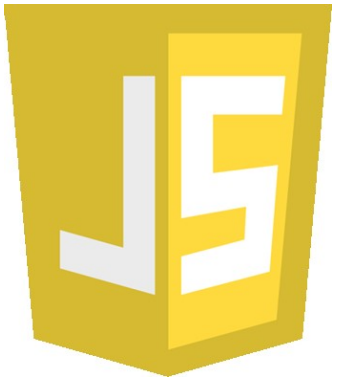
Javascript Training



Case 8: AMD

Case 8

AMD

- Introductie AMD
  - define
  - require
- Lazy loading met AMD
- Minification met AMD

Case 8

Introductie AMD

Current [edit]

- Modules/1.0 (Superseded by Modules/1.1)
- Modules/1.1
- Modules/1.1.1
- Packages/1.0
- Promises/B
- Promises/C
- System/1.0

Proposals [edit]

- Binary/B
- Binary/F
- Console
- Encodings/A
- Filesystem/A
- Filesystem/A/0
- Modules/Async/A
- Modules/Transport/B
- Packages/1.1
- Packages/Mappings
- Unit Testing/1.0

Case 8

Introductie AMD

- Asynchronous Module Definition
  - Module pattern, maar dan async
  - Asynchroon inladen van javascript
- Definieren van private modules met publieke API's
  - Modules kunnen gebruik maken van andere modules (require)
- Kan een stuk code hell oplossen
  - Biedt mogelijkheden tot structurering javascript sources
- Implementatie gebundeld bij Play2 Framework (RequireJS)

Case 8

Module pattern VS AMD

```
// Module definitie

(function(/** externe dependencies voor intern gebruik */){

    // Inner module

    return {
        // publieke API
    }

}(/* externe dependencies */));
```

```
// Asynchronous module definitie

define([/* externe dependency */], function (/* externe dependencies voor intern gebruik */){

    // Inner module

    return {
        // publieke API
    }

});
```

Case 8

AMD – Define VS Require

```
// Asynchronous module definitie

define([/* externe dependency */], function (/* externe dependencies voor intern gebruik */){

    // Inner module

    return {
        // publieke API
    }

});

// Geen module, maar code die uitgevoerd moet worden na laden dependencies

require([/* externe dependency */], function (/* externe dependencies voor intern gebruik */){
    // Doet iets met dependencies
});
```

Case 8

AMD – Define VS Require

```
// Asynchronous module definitie

define(function (require){
    var depA = require(/* externe dependency */);

    // Inner module

    return {
        // publieke API
    }

});
```

Case 8

AMD – Opzet

```
define(function () {

    ...

    return {
        enableLogging: function() {
            LogUtility.LOGGING_ENABLED = true;
        },
        disableLogging: function() {
            LogUtility.LOGGING_ENABLED = false;
        },
        isLoggingEnabled: function() {
            return LogUtility.LOGGING_ENABLED;
        },
        log: function(message) {
            logUtilityInstance.log(message);
        }
    }

});

define(['utilities/LogModule', 'services/MyRestService'], function (LogModule, MyRestService){

    // Inner module

    return {
        // publieke API
    }

});
```

Case 8

AMD – Lazy Loading

```
// MyModule.js
define(['utilities/LogModule', 'services/MyRestService'], function (LogModule, MyRestService){

    ...

    return {
        "eventListenerDieAfgaatBijKlikkenOpButton": function() {
            require(['LazyLoadedModule'], function(LazyLoadedModule) {
                // Doe iets met LazyLoadedModule
            });
        }
    }

});

// init.js
require(["MyModule"], function (MyModuleAPI){
    document.querySelector("button").addEventListener("click", function() {
        MyModuleAPI.eventListenerDieAfgaatBijKlikkenOpButton()
    }, false);
});
```

Atos