
Javascript Training



Case 6: Inheritance & Scoping

Case 6

Inhoud

- ▶ Prototype
 - Class inheritance
 - Overridden methods
 - Aanroepen super constructor en methods
- ▶ Scoping

Case 4

Prototyping

Object.prototype

Key	Value
constructor	Object
toString	function() { [native code]}
toLocaleString	function() { [native code]}
valueOf	function() { [native code]}
hasOwnProperty	function() { [native code]}
isPrototypeOf	function() { [native code]}
propertyIsEnumerable	function() { [native code]}

Javascript

Prototype – Class inheritance

Object.prototype

Key	Value
toString	function() { [native code]}

Parent.prototype

Key	Value
toString	function() { /* iets zinvols */ }
parentProperty	null

SubParent.prototype

Key	Value
constructor	SubParent

Javascript

Prototype – Class inheritance

```
var Parent = function(parentProperty) {  
    this.parentProperty = parentProperty;  
}
```

```
var SubParent = function() {  
    Parent.call(this, "parentPropertyWaarde");  
}
```

```
SubParent.prototype = Object.create(Parent.prototype);  
SubParent.prototype.constructor = SubParent;
```

```
var subInstance = new SubParent();  
subInstance.parentProperty === "parentPropertyWaarde";
```

Javascript

Prototype – Class inheritance

```
var Parent = function() {}
```

```
Parent.prototype.logNaarConsole = function(){  
    console.log("Ik ben de parentMethode");  
};
```

```
var SubParent = function() {  
    this.logNaarConsole();  
}
```

```
SubParent.prototype = Object.create(Parent.prototype);  
SubParent.prototype.constructor = SubParent;
```

```
var subInstance = new SubParent(); // → logregel op de console
```

Javascript

Prototype – Overridden method

```
var Parent = function() {}
```

```
Parent.prototype.logNaarConsole = function() {  
    console.log("Ik ben de parentMethode");  
};
```

```
var SubParent = function() {  
    this.logNaarConsole();  
}
```

```
SubParent.prototype = Object.create(Parent.prototype);  
SubParent.prototype.constructor = SubParent;
```

```
SubParent.prototype.logNaarConsole = function() {  
    Parent.prototype.logNaarConsole.call(this);  
    console.log("Ik ben de subMethode");  
}
```

```
var subInstance = new SubParent(); // → logregel op de console
```

Javascript

Scoping

► Global object



```
var mijnVar = "waarde";  
window.mijnVar === mijnVar;
```

```
function mijnFunctie() {  
    var mijnFunctieVar = "waarde"  
    mijnGlobaleFunctieVar = "waarde"; // window.mijnGlobaleFunctieVar = "waarde"  
}
```

```
mijnFunctie();
```

```
window.mijnFunctieVar === undefined  
window.mijnGlobaleFunctieVar === "waarde"  
window.mijnFunctie === undefined
```


Javascript

Scoping

```
var MyView = function() {};
```

```
MyView.prototype.log = function(message) {  
    console.log(message);  
};
```

```
MyView.prototype.renderGebruikers = function(gebruikers) {  
    gebruikers.forEach(function(gebruiker) {  
        this.log(gebruiker);  
    });  
};
```

Javascript

Scoping

```
var MyView = function() {};
```

```
MyView.prototype.log = function(message) {  
    console.log(message);  
};
```

```
MyView.prototype.renderGebruikers = function(gebruikers) {  
    var _this = this;  
    gebruikers.forEach(function(gebruiker) {  
        _this.log(gebruiker);  
    });  
};
```

Javascript

Scoping – Module Pattern

```
var LogUtility = function() { }  
LogUtility.LOGGING_ENABLED = true;  
  
LogUtility.prototype.log = function(message) {  
    if (LogUtility.LOGGING_ENABLED && LogUtility.isConsoleAvailable()) {  
        console.log(message);  
    }  
}  
  
LogUtility.prototype.isConsoleAvailable = function() {  
    return console !== undefined && console !== null;  
}
```

Javascript

Scoping – Module Pattern

```
// self invoking function (module)

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

    // Inner module

    return {
        // publieke API
    }

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

```

var LogModule =
  (function(console) {
    var LogUtility = function() { }
    LogUtility.LOGGING_ENABLED = true;

    LogUtility.prototype.log = function(message) {
      if (LogUtility.LOGGING_ENABLED && LogUtility.isConsoleAvailable()) {
        console.log(message);
      }
    }

    LogUtility.prototype.isConsoleAvailable = function() {
      return console !== undefined && console !== null;
    }

    var logUtilityInstance = new LogUtility();

    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);
      }
    }
  })(console);

```

```

var LogModule =
(function(logObject) {
    var LogUtility = function() { }
    LogUtility.LOGGING_ENABLED = true;

    LogUtility.prototype.log = function(message) {
        if (LogUtility.LOGGING_ENABLED && LogUtility.isLogObjectAvailable()) {
            logObject.log(message);
        }
    }

    LogUtility.prototype.isLogObjectAvailable = function() {
        return logObject !== undefined
            && logObject !== null
            && logObject.log !== undefined;
    }
    var logUtilityInstance = new LogUtility();

    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);
        }
    }
})(console);

```