Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

Speels Javascript dat voldoet aan ontwerpprincipes

Harrie Passier, Sylvia Stuurman en Harold Pootjes

Faculteit Management, Science and Technology
Open Universiteit Nederland

Inleiding

Zonder creativiteit geen innovatieve ict-systemen. Creativiteit kan ontstaan door principes los te laten. Soms krijgt creativiteit juist een kans door ontwerpprincipes te volgen. In deze bijdrage geven we hiervan een voorbeeld.
JavaScript is een programmeertaal die bevordelijk is voor de creativiteit, omdat je er webapplicaties mee kunt programmeren en het resultaat van de code direct te bekijken is in de browser. JavaScript kent aan de ene kant veel interessante aspecten, maar biedt aan de andere kant de mogelijkheid om erg 'lelijk' te programmeren,. Zo is JavaScript een object georiënteerde taal, gebaseerd op prototypes, en kunnen we in Javascript in een functionele stijl programmeren. JavaScript biedt daardoor de mogelijkheid studenten in een vroeg stadium kennis te laten maken met verschillende stijlen van programmeren en concepten van programmeertalen. Omdat Javascript een taal is waarmee makkelijk slecht geprogrammeerd kan worden, zijn ontwerpprincipes en richtlijnen nodig om te leren werken met deze taal.
Wij hebben een methode ontwikkeld, bestaande uit een stappenplan en richtlijnen per stap, waarmee op methodische wijze functionaliteit kan worden ontwikkeld voor form-validatie, iets dat nodig is in web applicaties. De methode maakt gebruik van de bewezen didactische aanpak 4C/ID [1] en bestaat onder andere uit de onderdelen taakbeschrijving, ondersteunende informatie en procesbeschrijving. We bespreken deze onderdelen kort en behandelen vervolgens kort een casus.

Taakbeschrijving

De taakbeschrijving beschrijft de taak in termen van requirements, in dit geval het valideren van data die een gebruiker invoert. Voorbeelden zijn de vraag of een ingevoerde postcode voldoet aan de vereiste structuur of de vraag of de combinatie van een gebruikersnaam en wachtwoord klopt. In onze aanpak onderscheiden we vier dimensies: autorisatie van de persoon die data invoert (betreft het de juiste persoon), tijdigheid van data-invoer (wordt de data op het juiste moment ingevoerd), compleetheid (zijn alle vereiste velden ingevuld) en correctheid (is wat is ingevuld juist) van de data die is ingevoerd. Omdat controle op autorisatie en tijdigheid op de server plaatsvindt, focussen wij op de aspecten compleetheid en correctheid. Compleetheid is alleen een check op aanwezigheid van data. Bij correctheid maken we onder andere onderscheid tussen structuur (bijvoorbeeld een postcode) en bereik (bijvoorbeeld een leeftijd tussen grenzen).

Ondersteunende informatie

Ondersteunende informatie beschrijft het domein van de taak en hoe kan worden bepaald of een oplossing acceptabel is. Hier maken we onderscheid tussen algemene ontwerpprincipes (bijvoorbeeld abstractie, koppeling, cohesie, decompositie en modularisatie) en specifieke principes voor form-validatie (bijvoorbeeld server-side validatie is verplicht, voeg zoveel validatie toe als noodzakelijk is, val de gebruiker niet lastig met interne representatie – zowel 1234aa als 1234AA zijn goed –, en streef herbruikbaarheid en uitbreidbaarheid van de software na.). Verder onderkennen we algemene richtlijnen

voor het programmeren in JavaScript (bijvoorbeeld: begin met het declareren van constanten en variabelen).

Procesbeschrijving

De procesbeschrijving beschrijft de stappen, de volgorde van de stappen en per stap welke van de principes of richtlijnen moeten/kunnen worden toegepast. Voor form-validatie wordt eerst vastgesteld welke data moet worden ingevoerd en hoe deze te specificeren (bijvoorbeeld eisen aan structuur en bereik) en hoe de gebruiker het best kan worden gestuurd bij het invullen van het formulier (bijvoorbeeld door het groeperen van invoervelden en het vaststellen van label- en placeholder-teksten) . Hierna wordt het formulier gerealiseerd door juiste HTML5 elementen te kiezen.
Vervolgens wordt de domeinlaag ontworpen en geïmplementeerd. Dit gebeurt eerst door één eventhandler te realiseren die controleer of alle velden die verplicht invoer moeten bevatten deze ook bevatten (compleetheid). Daarnaast wordt elk input-element waarvoor correctheidseisen gelden voorzien van een eventhandler die hierop controleert (correctheid). Deze eerste implementatie voldoet meestal niet aan de algemene ontwerpprincipes. Zo zal er vaak sprake zijn van redundante code en zal de code slecht uitbreidbaar zijn.
Een belangrijke vervolgstap is het refactoren van dit eerste ontwerp en implementatie volgens gegeven richtlijnen en ontwerpprincipes. Door een bepaalde volgorde in het toepassen van richtlijnen en ontwerpprincipes aan te brengen, wordt de kans groter dat een betere eindoplossing wordt gevonden. Het refactoren moet meestal in een aantal slagen plaatsvinden totdat een oplossing is gevonden die voldoet aan de algemene als ook aan de specifieke ontwerpprincipes.
In een laatste stap wordt het ontwerp en implementatie geëvalueerd en van documentatie voorzien.

Casus

De methode is toegepast op een casus, namelijk een formulier met twee inputvelden (een postcode en een aantal jaren met een bovengrens). In een aantal refactor slagen, waarbij met name redundantie in code, mate van uitbreidbaarheid en abstractie leidend zijn geweest, is toegewerkt naar een algemeen toepasbare module voor form-validatie met een eenvoudige interface die uit slechts twee methoden bestaat, namelijk isCorrect en isComplete. Omdat de HTML input elementen alle informatie bevatten die nodig is voor de validatie (zowel de ingevoerde waarde als de eisen die gesteld worden in de vorm van attributen als min, max en pattern) worden deze als geheel als parameter doorgegeven aan beide methoden. De terugkeerwaarde is een informatieobject dat de feedback bevat.
De casus laat zien dat het ontwerpen en implementeren van een module voor form-validatie zeker geen algoritmische activiteit is (succes is niet gegarandeerd), maar dat het proces en de richtlijnen en ontwerpprincipes per stap creativiteit op een juiste momenten stimuleert.
De methode en de casus zijn beschreven in een technisch rapport [2].

Literartuur

1. Jeroen J.G. Van Merriënboer, Richard E. Clark, and Marcel B.M. De Croock, Blueprints for complex learning: The 4c/id-model. Educational Technology Research and Development, 50(2):39_61, 2002.
2. Harrie.J.M. Passier, and Sylvia Stuurman, and Harold Pootjes, Beautiful JavaScript: How to create good and elegant code (Technical Report), Faculty of Management, Science and Technology, department of Computer science, Open Universiteit Nederland