CODING CONVENTION

S3-DB04 Groep 2

Inhoud

Java coding conventions

BRON

Class namen

Class namen moeten zelfstandige naamwoorden zijn, in combinatie met de eerste letter van elk intern woord met een hoofdletter. Probeer de namen van je klassen eenvoudig en beschrijvend te houden.

Voorbeeld:

```
class User;
class ImageSprite;
```

Class methoden

Methoden moeten werkwoorden zijn, in hoofdletters met de eerste letter in kleine letters, waarbij de eerste letter van elk intern woord met een hoofdletter wordt geschreven.

Voorbeeld:

```
run();
runFast();
getBackground();
```

Class variabelen

Namen van variabelen moeten kort maar zinvol zijn. De keuze van een variabelenaam moet een geheugensteuntje zijn, dat wil zeggen, als andere developers het lezen ze kunnen snappen waar het voor dient. Variabelenamen van één karakter moeten worden vermeden, behalve voor tijdelijke "wegwerpvariabelen". Gebruikelijke namen voor tijdelijke variabelen zijn i, j, k, m en n voor gehele getallen; c, d en e voor tekens.

Voorbeeld:

```
float myWidth;

String myName;
```

Constanten

Constanten variabelen moeten allemaal in hoofdleters zijn en met _ (laag streepje) gescheiden zijn.

Voorbeeld:

```
static final int MIN_WIDTH = 4;
static final int MAX_WIDTH = 999;
static final int GET_THE_CPU = 1;
```

Getter & Setter

Variabelen binnen een class horen private te zijn, dat betekent dat je buiten de class niet direct een variabele kan opvragen of aanpassen. Het opvragen en aanpassen doe je doormiddel van een getter en een setter methode.

Voorbeeld:

```
public class SimpleGetterAndSetter {

   private int number;


   public int getNumber() {

      return this.number;

   }


   public void setNumber(int num) {

      this.number = num;

   }

}
```

Vue coding conventions

BRON

Componentnamen

Componentnamen moeten altijd uit meerdere woorden bestaan, behalve voor root-app-componenten en ingebouwde componenten die door Vue worden geleverd, zoals <transition> of <component>. Dit voorkomt conflicten met bestaande en toekomstige HTML-elementen, aangezien alle HTML-elementen uit één woord bestaan.

Voorbeelden:

```
app.component('todo-item', {
})
```

```
export default {
  name: 'TodoItem',
}
```

Component Style scoping

In Vue kan je het bereik van een style van een component limiteren door ‘scoped’ toe te voegen aan de style attribute, dit is aangeraden omdat anders andere componenten de style zullen overnemen. Met scoped blijft de style binnen de component.

Voorbeeld:

```
<template>
  <button class="button button-close">×</button>
</template>

<!-- Using the `scoped` attribute -->
<style scoped>
.button {
  border: none;
  border-radius: 2px;
}

.button-close {
  background-color: red;
}
</style>
```

Private property names

Gebruik module scoping om privéfuncties van buitenaf ontoegankelijk te houden. Als dat niet mogelijk is, gebruik dan altijd het voorvoegsel $_ voor aangepaste privé-eigenschappen in een plug-in, mixin, enz. die niet als openbare API moeten worden beschouwd. Om conflicten met code door andere auteurs te voorkomen, voegt u ook een benoemd bereik toe (bijv. $_yourPluginName_).

Voorbeeld:

```
const myGreatMixin = {
  // ...
  methods: {
    $_myGreatMixin_update() {
      // ...
    }
  }
}


// Even better!
const myGreatMixin = {
  // ...
  methods: {
    publicMethod() {
      // ...
      myPrivateFunction()
    }
  }
}

function myPrivateFunction() {
  // ...
}

export default myGreatMixin
```

API conventions

BRON

Hou het simple

Bij het maken van een endpoint in de API, hou het simpel met urls, maak het leesbaar voor andere developers, zodat het geen zoek werk wordt naar de juiste endpoint. Voor RESTful URI zijn zelfstandige naamwoorden beter dan werkwoorden (werkwoorden worden afgeraden). Als vuistregel moet uw URI verwijzen naar een ding (een zelfstandig naamwoord) en niet naar een actie (werkwoord). Gebruiker meervoudige zelfstandige naamwoorden voor naamgeving.

Voorbeeld:

/users/12345 <- Duidelijk leesbaar voor een developer

/api?type=user&id=12345 <- Niet duidelijk leesbaar voor een developer

HTTP methodes

GET

Gebruik get voor het ophalen van informatie van de API (Read Operation)

POST

Gebruik post voor het versturen/aanmaken van data in de API (Write Operation)

Voorbeelden:

HTTP GET http://www.javadevjournal.com/orders //Get all orders

HTTP POST http://www.javadevjournal.com/users //Registers a customer with form data ofcourse