	**Hogeschool PXL - DIGITAL** Academiejaar 2019-2020 EXAMEN Web Scripting

Vak	
Resultaat	/20
Periode	1e zit
Datum	02/06/2019
Tijdstip	8:30
Klassen	1TIW A
Lectoren	Luc Doumen

Studentengegevens	
Naam student	
Voornaam student	
Klas	1 TIW A
Lector	
Lokaal	

Samenstelling bundel			
Onderdelen (*)	Examen		
Inhoud	3 opdrachten (examenvragen)		
Pagina's	11 pagina's		
Puntenverdeling	Oef1 : 10ptn	Oef2 : 15ptn	Oef3 : 20ptn
Digitaal beginbestand	ja, via blackboard		
Digitale indiening	ja, via e-mail	luc.doumen@pxl.be	
Toegelaten hulpmiddelen:			
* rekenmachine	ja		
* laptop	ja		
* internet	ja		
* cursusmateriaal	ja		
MD5Hash:			
Aanvang examen: ***start vrijdag 29 mei :*** *17:00* ***einde zondag 31 mei:*** *23:59*			

V O O R W O O R D

- Je krijgt inputbestanden toegestuurd via blackboard bij de aanvang van het examen (29 mei om 17:00u)
- Van deze bestanden moet je gebruik maken om de opdrachten op te lossen

- Ten laatste 31 mei om 23:59u stuur je per e-mail, **met leesbevestiging**, je oplossingen door. Bestemming luc.doumen@pxl.be
  De naam van het af te geven bestand moet als naamgeving van de vorm zijn:
  - naam_voornaam_webscripting.zip

- Controleer dus goed na het maken van de .zip of er alles inzit en dat je het juiste bestand doorstuurt

- Plaats in elk .js-bestand je naam en voornaam in commentaar
- **Alle code die je schrijft** komt in een daarvoor reeds **voorzien .js-bestand**
  Het spreekt voor zich dat, ter evaluatie, enkel maar naar die code wordt gekeken
- Bij de beoordeling wordt er gekeken naar werkende (onder)delen in het programma. Dus niet-werkende code wordt **NIET** beoordeeld

VEEL SUCCES !!!

Oefening 1 : Paardenwedstrijd **Ptn: ….. / 10**

De oplossing van deze opdracht moet in de map oef1 worden opgeslagen.

Deze oefening wordt uitgevoerd in de console via Node.js. Vermijd dubbele code.

Het is de bedoeling dat je in dit programma 4 klassen maakt, namelijk Paard, Ruiter, Wedstrijd en InputError.

Een wedstrijd bevat deelnemers (paard met ruiter).

Wanneer de wedstrijd start zal er random bepaald worden welk paard van de deelnemers een stap mag vooruit zetten. Dit zal via een for-lus gebeuren die 100x zal worden doorlopen.

Aan het einde van de wedstrijd wordt een overzicht weergegeven hoeveel stappen door elk paard is gelopen in de wedstrijd.

Je schrijft op basis van onderstaande opdracht al je javascript code in het bestand **oef1.js**.

Class InputError

- Maak de **ES6-klasse InputError**. Deze is afgeleid van de klasse Error

Class Ruiter

- De membervariabelen in deze klasse zijn _naam en _voornaam. Deze heeft een constructor met waarden voor de velden naam en voornaam.
- Voorzie getters voor _naam en _voornaam.
- Via de functie **toString**() wordt een string representatie van de ruiter gegenereerd. Van elke ruiter wordt de naam en de voornaam gegenereerd.
  Voorbeeld **toString()** : van Asten Leopold

Class Paard

- De membervariabelen in deze klasse zijn _naam, _ras, _telOp en _ruiter. Deze heeft een constructor met waarden voor de velden naam, ras, ruiter en telOp.
  - Ruiter moet een Ruiterobject zijn (hint: instanceof). Indien dit niet het geval is, wordt er een InputError “Er bestaat geen geldig RuiterObject” opgeworpen.
- Voorzie getters voor naam, ras, telOp en ruiter.
- Voorzie een setter voor telOp.
  - telOp moet een integer zijn (hint: **Number.isInteger).** Indien dit niet het geval is, wordt er een InputError “stappenteller moet numeriek zijn” opgeworpen.
- Via de functie **toString**() wordt een string representatie van de ruiter en het paard gegenereerd. Van elke ruiter wordt de naam en de voornaam gegenereerd, tezamen met het paard waarmee deze zal deelnemen aan de wedstrijd.
  Voorbeeld **toString()** :
  Ruiter : van Asten Leopold met paard : Amerigo, (ras) Appaloosa - # Stappen: 0

Class Wedstrijd

- De membervariabelen in deze klasse zijn _datum, _circuit, _deelnemers (=array). Deze heeft een constructor met waarden voor de velden datum, circuit. Binnen de constructor wordt het veld _deelnemers gelijkgesteld aan een lege array.
    - datum moet een Datumobject (hint: typeOf datum == 'object') zijn. Indien dit niet het geval is, wordt een InputError "foutieve datum" opgeworpen.
    - circuit moet een string (hint: typeOf circuit == 'string') zijn. Indien dit niet het geval is, wordt een InputError ('circuit is geen alfanumerische waarde' opgeworpen
- Voorzie getters voor datum, circuit en deelnemers (array).
- Voorzie een setter voor deelnemers (array). Deze is nodig om een object Paard toe te voegen aan de deelnemerslijst
- Via de functie **toString()** wordt een string-representatie van de wedstrijd gegenereerd.
  Voor de wedstrijd wordt de wedstrijddatum en het circuit gegenereerd.
  Voorbeeld toString(): Wedstrijd : (4/6/2020 - Hippodroom : Les Tilleuls (Kortijk))

Het hoofdprogramma

Je start met het maken van een wedstrijdObject, gevolgd door een ruiterObject, daarna een paardObject. Je voegt de ruiter toe aan het paard. Het paard wordt toegevoegd aan de array _deelnemers van het wedstrijdObject.

Onderstaande code is in het script reeds als testcode voorzien:

```
//Definieer wedstrijd

  wedstrijd = new Wedstrijd(new Date(2020,6,2),'Les Tilleuls (Kortijk)');

//Definieer paard, wijs ruiter aan paard toe, wijs paard aan deelnemerslijst toe

  ruiter = new Ruiter('van Asten', 'Leopold');
  paard = new Paard('Amerigo', 'Appaloosa', ruiter,0);
  wedstrijd.deelnemers.push(paard);

  ruiter = new Ruiter('Bartels', 'Tineke');
  paard = new Paard('Brego', 'Asturcion', ruiter,0);
  wedstrijd.deelnemers.push(paard);

  ruiter = new Ruiter('Brink', 'Jan');
  paard = new Paard('Huaso', 'Haflinger', ruiter,0);
  wedstrijd.deelnemers.push(paard);

  ruiter = new Ruiter('Capellmann', 'Nadine');
  paard = new Paard('Pegasus', 'connemara', ruiter,0);
  wedstrijd.deelnemers.push(paard);
```

Je schrijft **wel** javascript code in onderstaande blok:

// Start wedstrijd

- Een for-lus die je 100x laat doorlopen.
- Per iteratie bepaal je random (hint: *Math*.random()) welk object uit je deelnemerslijst (=paardObject) een stap voorwaarts zet (hint: telOp + 1)

// Toon OUTPUT (=klassement)

- Schrijf hier de code voor het tonen van de output (=klassement)
- **Een mogelijk voorbeeld van de output ziet eruit als volgt:**

  Wedstrijd : (4/6/2020 - Hippodroom : Les Tilleuls (Kortijk))

  Ruiter : van Asten Leopold met paard : Amerigo, (ras) Appaloosa - # Stappen: 23

  Ruiter : Bartels Tineke met paard : Brego, (ras) Asturcion - # Stappen: 24

  Ruiter : Brink Jan met paard : Huaso, (ras) Haflinger - # Stappen: 21

  Ruiter : Capellmann Nadine met paard : Pegasus, (ras) connemara - # Stappen: 32

  opmerking: som van # stappen moet 100 zijn

Oefening 2 : Getallen vormen .... / 15

De oplossing van deze opdracht moet in de map oef2 worden opgeslagen.

Je schrijft op basis van onderstaande opdracht al je javascript code in het bestand **calculator.js**.

Je mag in deze oefening geen gebruik maken van .innerHTML en innerText. Indien je dit wel doet worden de punten van deze oefening gehalveerd.

Je vertrekt van onderstaande code voor calculator.html en calculator.js.

Caculator.html

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <title>Getallen vormen en tonen</title>

   <link rel="stylesheet" type="text/css" href="./css/calculator.css">

   <script src="./js/calculator.js"></script>

</head>

<body>

   <H1>G E T A L E N <span>  </span>V O R M E N</H1>

   <output id="output_calculator"></output>

   <output id="output_results"></output>

</body>

</html>
```

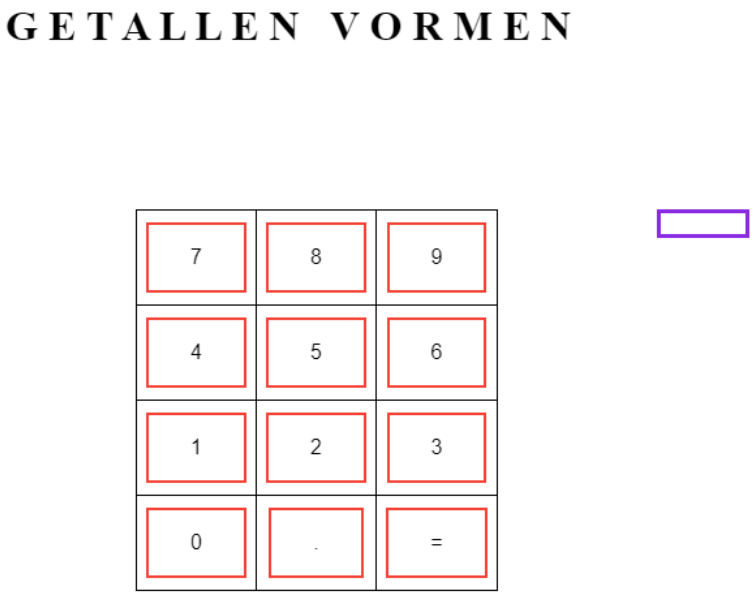
Caculator.js

```
let array_data = [[7,8,9],[4,5,6],[1,2,3],[0,".","="]];

const handleLoad = () => { }

window.addEventListener("load", handleLoad);
```

Er is een stylesheet calculator.css voorzien waarvan we de code niet oplijsten.

De code van calculator.html en calculator.css laat je verder **ongewijzigd**.

Na het oplossen van de opdracht wordt onderstaande (begin)toestand getoond in de browser:



- Het cijferbord wordt in de output-tag “output_calculator” getoond.
- Het lege paarse kader komt naast het cijferbord in de output-tag “output_results”. De bestaande css-code regelt dit automatisch.

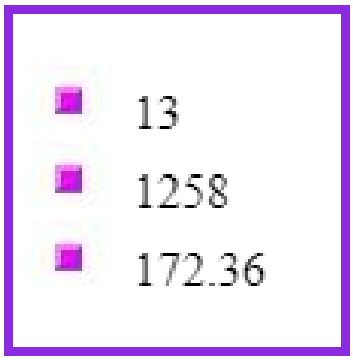
Het cijferbord

Het is de bedoeling dat je op een dynamische manier een tabel maakt van 4 rijen en 3 kolommen waarin elke cel een **button** voorstelt. De labels van iedere button vind je terug in de 2D array array_data in het bestand calculator.js. Elke button moet ook voorzien worden van de ccs class .button en css class .button_x.

```
let array_data = [[7,8,9],[4,5,6],[1,2,3],[0,".","="]];
```

De (lege) paarse kader

Hier is het de bedoeling dat je een **ongeordende lijst** laat verschijnen met daarin allemaal **listitems** die getallen voorstellen en die door het cijferbord worden samengesteld.



Werking van het programma

Getallen worden samengesteld door herhaaldelijk op de buttons te klikken. In een variabele wordt telkens achteraan het cijfer toegevoegd en waarvan die waarde gelijk is aan de label van de gekozen button.

Als er op het gelijkheidsteken (=) wordt gedrukt, wordt het getal getoond als een **nieuw listitem** in de **ongeordende lijst**.

Voorbeeld

Drukken op de button met label “1” en daarna drukken op de button met label ”3”, gevolgd door een druk op de button met het label “=” toont in de paarse kader een nieuw listitem met als waarde 13

Wanneer je vervolgens weer verder gaat met klikken op de buttons kan je nieuwe cijfers vormen en ze tonen in het kader.

Oefening 3 : Overzicht transacties (FETCH) …. / 20

De oplossing van deze opdracht moet in de map oef3 worden opgeslagen.

Je schrijft op basis van onderstaande opdracht al je javascript code in het bestand **transacties.js**.

Je mag in deze oefening geen gebruik maken van .innerHTML en innerText. Indien je dit wel doet worden de punten van deze oefening gehalveerd.

Je vertrekt van onderstaande code voor **transacties.html** en **transacties.js**.

transacties.html

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" type="text/css" href="./css/transacties.css">
   <script src="./js/transacties.js"></script>
</head>
<body>    <h2>Overzicht transacties</h2>
   <label for="transactie">Kies een mutatiesoort:</label>
   <select id="transactie">
      <option value="Geen" DEFAULT>Geen selectie</option>
      <option value="Overschrijving">Overschrijving</option>
      <option value="Betaalautomaat">Betaalautomaat</option>
      <option value="Internetbankieren">Internetbankieren</option>
   </select>
   <br/><br/><br/><output id="output"></div>
</body>
</html>
```

transacties.js

```
let hoofding = ["Datum", "Naam", "Account", "Bedrag", "Mutatiesoort", "Mededeling"];

const handleLoad = () => {

}

window.addEventListener("load", handleLoad);
```

Er is verder nog een stylesheet **transacties.css** en een .json **transacties.json** voorzien waarvan we de code niet oplijsten.

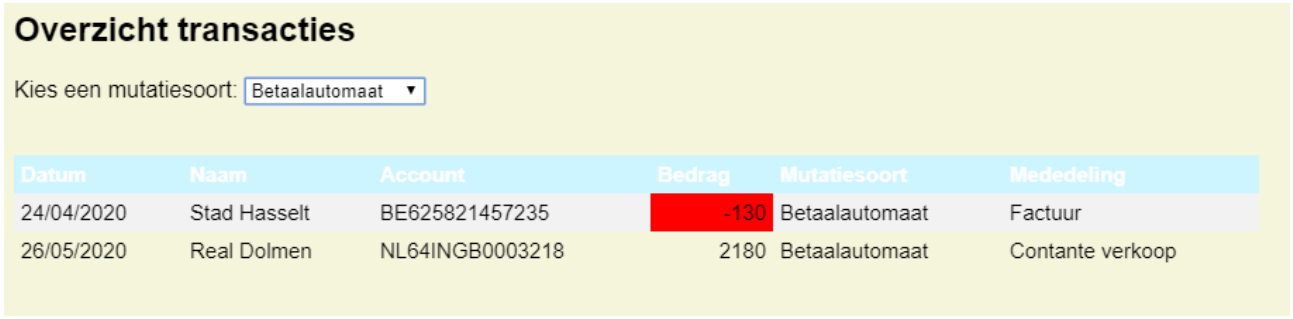
De code van **transacties.html, transacties.css** en **transacties.json** laat je **ongewijzigd**.

Na het oplossen van de opdracht wordt onderstaande (begin)toestand getoond in de browser:

Wanneer een waarde wordt gekozen in de keuzelijst moet er dynamisch een HTML-tabel gemaakt worden. De data die getoond wordt in de tabel is gefilterd, op basis van de gekozen waarde uit de keuzelijst en afkomstig van het bestand **transacties.json**.

Een voorbeeld :

Onderstaand een dynamische tabel met daarin de data, geselecteerd op basis van de waarde “betaalautomaat” uit de keuzelijst en afkomstig van het bestand **transacties.json**.



- De gehele boomstructuur van de HTML tabel moet gekoppeld worden aan de output-tag “output” (zie transacties.html)
- De hoofding van de tabel maak je op basis van de data uit de array **hoofding** die voorzien is in het bestand **transacties.js**
- De data in de tabel is afkomstig van de .json **transacties.json**
  (HINT : Gebruik hiervoor de FETCH-methode)
- Opmaak van de tabel:
  - De data in de kolommen Datum, Naam, Account, Mutatiesoort en Mededeling moet links uitgelijnd worden
  - De data in de kolom bedrag wordt rechts uitgelijnd
  - Indien het bedrag een negatief getal is moet de cel een rode achtergrondkleur krijgen. (Er is in de .css een class ‘rood’ voorzien die je moet gebruiken)