

</> JAVASCRIPT · P5.JS

Tech Review Programming 101

Tien vragen over jouw eigen code

ICT | HOGESCHOOL VAN AMSTERDAM · CREATING TOMORROW

Aan het eind van deze sprint laat je tijdens de **Tech Review** in een kort gesprek zien dat je **je eigen code begrijpt**. Je docent stelt een aantal van onderstaande vragen en jij beantwoordt ze aan de hand van jouw eigen p5.js-sketch. Alle vragen staan hieronder, zodat je je goed kunt voorbereiden. Er zijn dus geen verrassingen. Je code hoeft niet perfect te zijn; je uitleg erover telt.

Zo werkt het gesprek

- Je hebt je project open in je IDE én je demo draait in de browser.
- Per vraag wijs je de plek in je code aan en leg je hardop uit wat daar gebeurt.
- Het gaat om begrip, niet om definities uit je hoofd leren: uitleggen in je eigen woorden is precies goed.
- Weet je iets even niet? Zeg dat, en laat zien hoe je het zou uitzoeken.

Zo bereid je je voor

- Loop je sketch regel voor regel door en vertel bij elke regel wat die doet.
- Oefen hardop, bijvoorbeeld met je peer-reviewer.
- Zoek bij elke vraag hieronder alvast het stukje code op waar je naar wijst.
- Kom je ergens niet uit? Markeer het en stel je vraag in de les.

Wat je in Portflow doet

VOORAF

Upload je repo als bewijs

Zet vóór de Tech Review een [link naar je Programming 101-repo](#) in Portflow. Je uploadt die onder **de Techneut** en koppelt het bewijs aan leeruitkomst **Ontwerpen & Realiseren**. Zonder die link kan het gesprek niet doorgaan.

NA AFLOOP

Zet de feedback erbij

Je schrijft de feedback van je docent zelf op als **comment** bij dat bewijs. Doe dit dezelfde dag en schrijf in volzinnen.

AFRONDEN

Koppelen en laten valideren

Koppel je comment aan de juiste **succescriteria** van Ontwerpen & Realiseren (zie hieronder) en stuur een **validatieverzoek** naar je docent. Zonder validatie is het een aantekening en telt het niet mee als bewijs.

Koppel je comment aan deze succescriteria van Ontwerpen & Realiseren

- 1 Je bouwt voor een eenvoudig probleem een oplossing in code die voldoet aan de gestelde eisen.
- 3 Je richt een ontwikkelomgeving in en gebruikt deze consistent bij het uitwerken van oefeningen en projectcode.
- 4 Je verdeelt je code over modules, schrijft functies met één taak, hergebruikt code om duplicatie te vermijden en levert een volledig werkend product en kunt dit toelichten.
- 8 Je test functionaliteit met hulp, documenteert problemen en past je oplossing aan op feedback van docenten of peers.

De tien vragen

1 OPBOUW P5.JS

Leg uit hoe jouw p5.js-programma is opgebouwd. Wat is het fundamentele verschil in werking tussen de **setup()**- en de **draw()**-functie in jouw code, en hoe vaak worden deze uitgevoerd?

```
// setup() / draw()
```

2 VARIABLEN

Wijs in jouw code een variabele aan die je hebt gedeclareerd met **let** of **const**. Wat is het datatype van de opgeslagen waarde? Waarom heb je in deze specifieke situatie voor **let** of juist **const** gekozen?

```
// let / const
```

3 COÖRDINATEN & BEWEGING

Het assenstelsel in p5.js werkt anders dan bij wiskunde: **(0,0)** zit linksboven. Laat in je code zien hoe je een object naar beneden of omhoog laat bewegen, en leg uit hoe de **Y-waarde** zich daarbij gedraagt.

```
// x, y
```

4 CONTROL FLOW

Wijs een **if**-constructie aan in je code (bijvoorbeeld voor een stuiterende bal). Welke vergelijingsoperatoren (zoals **>** en **<**) of logische operatoren (zoals **&&** en **||**) heb je gebruikt om te bepalen wanneer de actie wordt uitgevoerd?

```
// if ( ) { }
```

5 ARRAYS

Laat zien waar je een **array** hebt gebruikt om meerdere waarden op te slaan. Hoe vraag je in de code op hoeveel elementen erin zitten, en hoe heb je dynamisch een nieuw element toegevoegd?

```
// [ ].length
```

6 LOOPS

Wijs een **for**-loop aan in je code. Leg de drie onderdelen uit: de **initialisatie** (startwaarde), de **conditie** die bepaalt wanneer de loop stopt, en de **update** (het aanpassen van de teller).

```
// for (i = 0; ...)
```

7 FUNCTIES, PARAMETERS & RETURN

Wijs een zelfgeschreven functie aan (dus niet **setup** of **draw**). Welke **parameters** geef je mee als argumenten? Gebruikt jouw functie een **return** om een waarde terug te geven, en waarom wel of niet? Leg aan de hand van de parameters en de returnwaarde uit hoe de functie in de rest van je code gebruikt wordt.

```
// return
```

8 FUNCTIES & SCOPE

Wat is het verschil in zichtbaarheid tussen een variabele die je bovenaan je sketch hebt gedeclareerd (**globaal**) en een variabele binnen in een functie (**lokaal**)? Wat gebeurt er als je een lokale variabele buiten die functie probeert te gebruiken?

```
// { scope }
```

9 OPERATOREN & GELIJKHEID

Leg aan de hand van een voorbeeld in je code uit wat het verschil is tussen **=**, **==** en **===**. Waarom gebruiken we bij voorkeur **===**?

```
// = == ===
```

10 FOUTANALYSE & DEBUGGING

Stel dat een loop in jouw code vastloopt of een oneindige loop wordt. Welk specifiek onderdeel van de loop zou je als eerste inspecteren, en hoe pak je dit aan met **debugging**?

```
// console.log()
```