- **Explain the difference between the == operator and the === operator.**

Het verschil tussen de == en de ===-operator is dat met de eerste operator er alleen maar gekeken wordt naar de uiteindelijke waarde, en niet het type van het object. Dit is nodig, omdat sommige waardes (0, NaN, “”) in Javascript ook als ‘false’ worden gezien. Voorbeeld:

0 == false -> true
0 === false ->false

Dit maakt het mogelijk dat, als je het nodig hebt, je op de specifieke waarde kan controleren.

- **Explain what a closure is. (Note that JavaScript programs use closures very often.)**

Closure is de mogelijkheid om lokale variabelen te gebruiken in een functie, nadat de functie is aangeroepen. Namelijk, als je een functie aanmaakt in een functie, die gebruik maakt van lokale variabelen, dan blijven die variabelen actief. Het is een manier om variabelen mee te geven voor een functie, en daarna de functie vaker kunnen gebruiken met diezelfde variabele.

- **Explain what higher order functions are.**

Higher order functions zijn functies die gebruik maken van een andere functie. Soms als argument, en soms als een functie die ze teruggeven. Een goed voorbeeld hiervan is de map() functie van een array. De map-functie is een higher order functie, omdat je een functie meegeeft aan de map-call met als argument de waarde (of de positie en de waarde) om hier vervolgens iets mee te doen. Het is een hele krachtige functie van Javascript (maar bestaat ook in andere programmeertalen), die het makkelijk maakt duidelijke code te schrijven.

Explain what a query selector is and give an example line of JavaScript that uses a query selector.

Een Query Selector is een manier om via CSS-selectors duidelijk te maken naar welke elementen in de HTML-broncode wil refereren. Je kan zo eenvoudig door de structuur van een HTML-code heengaan, op basis van attributen, id's, en classes (op dezelfde manier als waarin je in een CSS-bestand naar je elementen refereert).

Een voorbeeld hiervan werd gebruik voor de opdracht:

```
ApplePie.creator = document.querySelector("#header > p").innerHTML;
```

Dit laat zien, dat ApplePie.creator de interne content wordt van het p-element direct onder een element met id 'header'. Voor de ingrediënten werkt het vergelijkbaar:

```
ApplePie.ingredients = Array.prototype.slice.call(document.querySelectorAll('#ingredient-list > ul > li')).map(function(element) {
        return element.innerHTML;
});
```

Hierbij wil ik niet één element (zoals bij querySelector()), maar alle elementen die aan de voorwaarde voldoen van #ingredient-list > ul > li. Ofwel, ieder <li> element dat onder een <ul> element zit, dat onder een element met id 'ingredient-list' zit. Het maakt het heel makkelijk om individuele elementen te gebruiken in Javascript/CSS-code.