Brainnest Homework / Terminology & Definitions (JavaScript_2)
Clémence TAN (group B)

1. Why is it important to write clean code ? **(pptx p-)**
=> It is important to write clean code because it will help better communicate with other colleagues who will be albe to understand faster what the content is all about when he or she or they will take over the code for the next coding tasks.

2. What is the difference between good comments and bad comments?
=> Definition-wise,

- good comments says about « **why** » the code is doing what it does
- bad comments says about « **what** » the code is doing

3. What is an array?
=> according to Aleksa's lesson via pptx document, an array:

- is not primitive and is resizable
- is a collection of items
- can contain a mix of different data types
- is accessible via integers as indexes, with the first element indexed as 0

4. What are arrays userful for?
=> It is useful for organizing and storing data :

- storing multiple pieces of data of the same type together
- storing as well a variety of data types : integers, floats, strings... etc
- implementing lists

5. How do you access an array element?
=> By using indexes inside square brackets, like : [0], [1], [2]...

6. How do you change an array element?
=> There are multiple ways to do it and this also depends on which element inside the array we want to change : add, remove, or replace...

- **.push()**
- **.pop()**
- **.shift()**
- **.unshift()**
- **.splice()** ... etc

7. What are some useful array properties?
=> they are :

- **length** (sets or returns the number of elements in an array)
- **prototype** (allows to add new properties and methods to an array object)

8. What are some useful array methods?
=> they are :

- **.filter()**
- **.find()**
- **.forEach()**
- **.indexOf()**
- **.join()**
- **.map()**
- **.pop()**
- **.push()**
- **.split()**
- **.sort()**
- **.splice()** … etc

9. What are loops useful for?
=> Loops are especially useful to perform repeated tasks based on a condtion, this helps not to write the same code multiple times.

10. What is the break statement?
=> It ends, breaks out or helps exit the current loop or swith.

11. What is the continue statement?
=> It makes an end with the current iteration and continues or moves on to the next iteration.

12. What is the DOM?
=> Document Object Model.

13. How do you target the nodes you want to work with?
=> The DOM (Document Object Model) is made out of nodes, and we target (select) the nodes by writing : document.**getElementsById**('idSelector') ; there are multiple ways to write the green part :

- **getElementsByTagName()**
- **getElementsByClassName()**
- **querySelector()**
- **querySelectorAll()**

14. How do you create an element in the DOM?
=> By creating a variable, as following :
```
const div = document.createElement("div");
```

15. How do you add an element to the DOM?
=> By creating first the element node as answered in the previous question, then append it to an existing element, like `div.appendChild(h1);` here we insert « h1 » into the « div ».

16. How do you remove an element from the DOM?
=> By writing **element.remove()**

17. How can you alter an element in the DOM?
=> When we need to add css properties, we can do, ie, for a div element :
```
div.style.backgroundColor = "blue";
```

18. When adding text to a DOM element, should you use textCoente or innerHTML?
=> Based on Aleksa's lesson, textContent is preferable than innerHTML, because innerHTML may create security risk if misused.

19. Where should you include your JavaScript tag in your HTML file when working with DOM nodes?
=> **<script></script>** should be place right before the very end of **</body>**, closing body tag.

20. How do « events » and « listeners » work?
=> « events » : they are triggered inside the browser window, and tend to be attached to a specific item
=> « listeners » : to react to an event, we add a listener (event handler) to it, written as :
```
let btn = document.querySelector("button");
btn.addEventListener("click", function(e){
  console.log(e);
  e.target.style.background = "blue";
})
```

21. What are three ways to use events in your code?
=> inline
=> using a property
=> using a listener

22. Why are event listeners the preferred way to handle events?
=> « event handler » : **only one event handler** for a specific event type, whereas,
=> « event listener » : **multiple event listeners** can be added to a specific event type

23. What are the benefits of using named functions in your listeners?
=> A named function helps the code be more **DRY** (Don't Repeat Yourself).
=> Will be easier to be removed if later we want to use **removeEventListener()**, as this will be impossible if the code were written with a anonymous function.

24. How do you attach listeners to groups of nodes?
=> To add an event listener to multiple elements, for example :
```
let elements = document.querySelectorAll(".button");
let clickEvent = () => {
  console.log("It is clicked.")
}
elements.forEach((item) =>{
  item.addEventListener("click", clickEvent)
})
```

25. What is the difference between the return values of querySelector and querySelectorAll?
=> vcvcwv

26. What does a « nodelist » contain?
=> **querySelector** : returns the first single element in the document that matches ;
=> **querySelectorAll** : returns a collection of all elements in the document that match

27. Explain the difference between « capture » and « bubbling ».
=> **event bubbling** : the event is first handled by the innermost element and then propagates towards the outer elements.
=> **event capturing** : the event is first handled by the outermost element and then propagates towards the inner elements.

28. What is the difference between objects and arrays?
=> **object** : is a collection of properties, and a property is an association between a name (key) and a value.

```
let object = {
  name: "appel",
  color: "red",
  category: "food"
}
```

=> **array** : a special type of object that represents a list of items.

```
let array = ["dog", "cat", "bird"];
```

29. How do you access object properties?
=> If taking the answer for the previous question as an example, if we want to access the color property of the object, we type : **object.color**