1. **List and explain the JavaScript methods that enable us to select elements from the DOM.**

JavaScript offers six methods that enable us to select an element from the DOM:

1. Using getElementById, you can find an element by its ID.

   In this method, we can return an element that its ID matches the passed string. The getElementById method is only available in the document object because the id must be unique, we don't need a different function. If the id is not defined or found, then the output for this method will be null. For example:

```
document.getElementById('card');
```

2. Using getElementsByTagName, you can find an element by its tag name, such as span or div. While the getElementById method is only available in the document object, the getElementsByTagName can be called on any element. If getElementsByTagName is called upon an element, then only children of the element are searched. For example:

```
let parent = document.getElementById('card');

let spans = card.getElementsByTagName(span);
```

   here the span inside the element will be returned.

3. Using getElementsByClassName, you can find an element by its class name.

   To look for multiple class names, we can input the class names separated by a space. For example:

```
let dogElements = document.getElementsByClassName("dog name");

let catElements = document.getElementsByClassName("caat name");
```

4. Using getElementsByName, you can find an element by its name attribute.

   This method retrieves a collection of elements that correspond to the name attribute's value, based on the string passed as an argument. This collection is a live NodeList. For example:

```
document.getElementByName("first-dog");
```

5. Using querySelector will return the first element that matches the passed selector.

   For example:

```
// using id selector
document.querySelector("#parent");

// using class selector
document.querySelector(".child"); //Returns first element in the document with class name

// using tagname
document.querySelector("div"); //Returns first div


// using tagname
document.querySelector("div span.child"); //Returns first span element with class name child, ins
ide a div
```

The querySelector function can be invoked on both the document and element objects. It returns null if no elements match the provided CSS selector. An error of type SyntaxError is thrown when the selector is not valid. To specify multiple selectors, one can separate them by commas.

6. Using querySelectorAll is an extension of querySelector method that returns all elements that match the given selector. For example:

```
document.querySelectorAll(".child");//returns all element with child class
```

Source: https://www.educative.io/answers/what-are-the-different-ways-to-select-dom-elements-in-javascript

2. **What is the document object? What are some other methods available on the document object that don't select elements from the DOM?**

The Document Object Model (DOM) serves as a data representation of a web document's structure and content. The purpose of the DOM is to provide a representation of a webpage that allows programs to modify trrrr structure, style, and content. This representation takes the form of nodes and objects, which enables programming languages to interact with the page.

Besides selecting elements from the DOM, the document object provides many other methods. Here are some examples:

1. document.createElement(name)- Creates a new HTML element.
2. parentNode.appendChild(node)- Creates a new text node.
3. document.open() - Opens a new empty document in the same window.
4. document.close() - Closes the current document after it has been opened with document.open().
5. EventTarget.addEventListener() to listen for button clicks.

6. document.removeEventListener() - Removes an event listener that was previously registered with document.addEventListener().
7. element.style.left – align the element to the left.

Source:

https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Model/Introduction