Javascript DOM

style:

WIth this property we could set the css of our HTML object easily, and it's used in this way:

```
object.style.cssProperty=value
```

The difference between getElementByTagName() and querySelectorAll()

The former one will return us a HTMLCollection that is an array-like object, so this object is not a really array that can only use the index and length property as the normal array, but the array method forEach. Instead, the latter one is an object of collection. It allows us to use the properties and the methods a normal array has includes the method forEach

The difference between childNodes and children

Both of them will return a array of the sub-objects in this object, but the former will take anything in it as an object of it, it means that the whitespace is also considered as a text node, but the latter won't, so it's why we use the latter more

parentElement

We use it's property to navigate the parent element of the object

previous(/next)sibling

With these two properties we could navigate the siblings, but lie the childNodes, these properties include the whitespace, to avoid that, we could use previous(/next)ElementSibling instead

textContent

This is a property taht offers us to get the text content of this object like it's name, and also with this property gives us a way to set the the value of the text cotent

Note: There is one thing we should take the attantion:both methods getElementByTagName and getElementByClassNamereturn a HTMLCollection, enven if there is only one value init, so when we need to set property to the object ,we should use the index to get a certain object first, but byId is not included, cause the particularity of Id

concept of classList

This is a property of the DOM object, with this property we could invoke the array methods:add(values) and remove(values),so use it we could have a better way to add and remove the class in this object

document.createElement(Element TagName),doucument.createTextNode('text'),object.appendChild(childElement)

Use these methods we could create the elements and fill the element then add it to a specific object

parentElement.insertBefore(element,location):

With this method, we could insert the element to the spefic location which is before the location object in the parentElement

parentElement.replaceChild(newElement,oldElement)

Use this method we could replace the initial childElement in the parentElement with the new one

prepend(Element)

With this method we could add the element to the head of the object

difference between createTextNode and innerText

From the side of the name, we could find that the former is creating a new object, but the latter is just eidtting the text inside the object, so the former is add the object generated by the former, and made it as a childElement

remove() and removeChild(child)

If we could create and add new element ,we could also remove that , so there goes two methods, the first one is invoked by the object itself, so when is happened, that object is removed.But the second is invoked by the parentElement, so the specific childElement inside this parentElement will be removed

innerHTML

Like the textContent, this property will return the whole html structure of this object. So it means if we set the value to it, we could also change the html structure of this object

addEventListner(event,callBack)

As we know, we could set a event in directly way like:

```
object.onEvent = function () {};
```

or we just use the listeners like this:

```
object.addEventListner(event,callBack/callback reference);
```

But in this way , we should notice one thing that we could use an annoymous function direstly in the listener, and also we could use a callback reference instead, but this reference should not include the parentheses, because if you add them, it will be invoked before you trigger that event.

nameInput.value()

Within the keyup event, we can use nameInput.value to get the current value we're inputting, and this value will always be updated after you entered

argument in callback

When we use the callback function in eventListener,we could set an argument in it, it canbe anything, but we used to set it as e or event, and this e represents the whole properties as an object, and we can use e.currentTarget to get the actual object

use this vs use event.currentTarget

As we know, "this" always represents the current object whatever in Java or javascript, so when we use it in this callback function at listener, we will find that is has the same function as e.currentTarget, but in fact , they get a little difference: when we use e.currentTarget, it can still work to point to the current object when function is at abbreviation mode, but "this" would not:

```
object.addEventListner('click', (event) => {
  console.log(event.currentTarget);
  console.log(this);
});
```

At this moment, currentTarget is still pointing to the object, but the key word "this" will point to the object Window

the difference between the currentTarget and target

When we are in the callback function at a listener, we could use e to invoke both target and currentTarget to get current object. The formmer one always refers to the element to which the event handler has been attach to, whatever this element has so many nested elements, they are all in this element, so they are one combined object, but the latter one indentifies the element on which the event occurs. So when an element has so many nested elements in it, this element will divide into so many layers of elements, only when you select one layer, that layer of element and the rest of the elements nested in that element will trigger the event.currentTarget.

the use of the propagation

We know that the propagation is a mechanism to propagate events, and it performs from the nested element to the parent element one by one, so we could use the propagation mechanism to achieve the same behavior we could not do when the element is not created before attach the event handler. For example:

We created container and there is nothing in it. Then we create a button element or anything else, then we set a event handler on this button element that makes that container genereate a new element in it that we wish

these new elements could attach a event handler.

At this time, we could attach the event handler to the container instead of attaching to the new element with the help of the propagation mechanism. Because when we attach the event handler to the container, we could use event.target to get the new element that makes the event occur, and then set this new element.

preventDefault() in form

The reason for creating a form is submitting the values we grabbed from the that, hoerver when we click on the submit button, the page will be sent to the target url imidiately, no matter whether the form is being filled out properly, so in order to prevent the default behavior we could use preventDefault() to prevent the default behavior until we want the default behavior happen

localStorage and sessionStorage

These two storage object are offered by browser itself, so we could find them in the debug console panel at application part. These two object provide the methods setItem(key,value), getItem(key),remove(key) and clear(). So we could notice that these values are matched with keys. Once you give the value to this key, the previous value will be overwrited. For the practical use of these methods , there will be skipped. But there is one thing we should take attention, that is is the difference between both two object. The former one will always exist at this page until you remove it, even when you comment the code you create it, and reopen the browser, but if you do the same thing to the latter one, it can only exists in this page, when you open the another page , it won't work

deeper understand for localStorage

We could set a sinlge localStorage so easily, but when we want store an array into localStorage, if we just keep it in that way, it won't work correctly. Because when you store the array in localStorage directly, this array will not be stored as a array anymore. So at this time, we should use the JSON.stringfy() to convert this array to a JSON string, then store it in the localStorage. After that, when we want to pick them out, cause they are now stored in type of json string, so we need to convert them back to the initial state, so we need to use JSON.parse() to parse it, then we could use them correctly. Finally, we still need to pay attention to something it's that after we store an array to the localStorage,cause it is not synchronized so we could method push to add the data into it each time we refresh the page:

```
let fruits;

if (localStorage.getItem('fruits')) {
  fruits = JSON.parse(localStorage.getItem('fruits'));
} else {
  fruits = [];
}
console.log(fruits);
fruits.push('Orange');
localStorage.setItem('fruits', JSON.stringify(fruits));
```

So after we use this module, we could add a new element 'Orange' into the fruits, so with the help of eventListener there will be an another useful use

setTimeout(function reference/call back,time)

We found that when we set a timeout, we could set a callback function or invoke the function reference, so there's a question, how do we pass the argument t0 this reference? To solve this, we need just set the arguments after the time, like this:

```
setTimeout(function referrnce,time,argument1,argument2...)
```

setInterval(function reference/call back)

With the extremely same similarities as setTimeout, we could use all the properties of that. The only differednce is the method in the interval will run repeatedly until we use the method clearInterval(interval Name) to end it. Of course we could also use clearTimeout to end the timeout too.

the difference between load and DOMContentLoaded

The former one will be fired when the whele page has loaded, inluding all the dependent resources such as stylesheet and images instead of the latter one is only take the DOM load into account

use scrollX and scrollY

We could use these two properties to get the pixels that we scrolled

innerHeright, innerWidth and getBoundClientRect()

These two properties belong to the object window, so they return the height and width of the window. As dor the method getBoundingClientRect is set to return the object that contains the real distance data of the actual element to the window

join(separator)

This method is fimilar to this method in Java, so with this method we could concat the data of an array together with the separator, if by default the separator is the coma, and we could set the separator to something else, like when we put an space in it , they will concat with space

dataset

It is a property that contains all the custom data, for example when we set a property 'data-name' in HTML tag, we could use dataset.name to get the value in this custom property.

call（）,apply() and bind()

The call method is used to change the pointer to keyword "this" , it means when you create an object and use it's reference to invoke an method of it, the keyword "this" references to the current instance, but with the

method call we could change the reference of "this", make it point to an another instance, and there is an example:

```
function person(name, age) {
  this.name = name;
  this.age = age;
  this.greet = function (sex) {
    console.log(
      `Hello, my name is ${this.name} and my age is ${this.age}, i'm a ${sex}`
    );
    return `Hello, my name is ${this.name} and my age is ${this.age}, i'm a
${sex}`;
  };
}
const jerry = new person('jerry', 20);
const tom = new person('tom', 22);
jerry.greet.call(tom, 'boy');
jerry.greet.apply(tom, ['boy']);
const message = jerry.greet.bind(tom, 'boy');
```

So we found when we use call or apply these kind of methods, we need just call the method reference insted of invoking it directly. and there's potint, there are some differences between call , apply and bind: The call and apply method will pass the arguments in list but the apply passes the arguments with object. Lastly the bind method has the difference to the others is that we can value the other variable with this method