Cheat Sheet – Events

Events are of course an integral part of modern applications. With Events, you can react to user actions and provide a reactive user experience.

The Event Interface

The Event Interface is implemented by a lot of DOM Events and it exposes useful properties and methods to work with Events.

With those methods and properties you can, for example, retrieve the target of the event or change its default behavior (e.g. prevent it from propagating).

Learn more about it here: https://developer.mozilla.org/en-US/docs/Web/API/Event

DOM / User Events

Amongst the most important Events you may listen to, are of course user Events triggered while interacting with the DOM.
For example, a user might click on a element. You can listen to such events and execute code whenever they get triggered/ fired.
Besides click events, there of course also possibilities to listen to a lot of other events.

Visit this link for a list of available events: https://developer.mozilla.org/en-US/docs/Web/Events

Event Handlers

Event Handlers are set up on the emitting element itself. For example you could set up a click listener like this:

```
document.querySelector('#container1').onclick =
function(event) {…}
```

Notice that the event object gets passed. This event object is available on each event you might listen to.

To remove an Event Handler, simply set it to null.

Event Listeners

The disadvantage of Event Handlers is, that you only may have one at a time. But sometimes you need more than one listener.

This is where Event Listeners come into play. They are set up differently and you may provide as many as you wish:

```
document.querySelector('#container1').addEventListener('click', listener1);
document.querySelector('#container1').addEventListener('click', listener1);
```

```
document.querySelector('#container1').addEventListener('click', listener1);

document.querySelector('#container2').addEventListener('click', listener2);

function listener1(event) {
    console.log('Listener 1 here');
}
function listener2(event) {
    console.log('Listener 2 here');
}
```

Event Listeners can of course also be removed

```
document.querySelector('#container2').removeEventListener('click', listener2);
```

Event Behavior

Events have some default behaviors. One very important one, is, that they propagate.
That's best explained as an example: If you have to elements which are nested inside each other, then you might want to listen to clicks on the inner one.
However, since Events propagate, a click on the inner one will also trigger all click listeners on the outer one.
This might not be the behavior you want and you can stop this from happening by calling stopPropagation() on the event object passed into the handler function.

```
function listenerInner(event) {
    event.stopPropagation();
}
```

Learn more about Events here: https://developer.mozilla.org/en-US/docs/Web/API/Event