Events

Events

Users interact with our pages

Interactions trigger events

Events can trigger functions

Events - Terminology

Listen for events

Handle events

Events are **triggered** or **fired**

What are some possible user interactions?

- Click
- Double click
- Hover
- Submit form
- Change value of input fields
- Scroll
- Tab focus changes
- Element focus changes
- Key presses
- Copying/pasting

- Orientation change
- Compass heading
- Accelerometer
- Battery status
- Light sensor

addEventListener

A function on an element

First argument is the type of event, a string

Second argument is a function, the event handler

```
element.addEventListener(“click”, event => {

});
```

addEventListener - click

```
const buttonElement = document.querySelector(“button”);


buttonElement.addEventListener(“click”, event => {
    console.log(“the button was clicked!”);
});
```

addEventListener - click

```
const headerElement = document.querySelector(“h1”);


headerElement.addEventListener(“click”, event => {
    console.log(“the header was clicked!”);
    console.log(event.target);
});
```

addEventListener - click

To the code editor and browser

addEventListener - input

```
const inputField = document.querySelector(“input”);


inputField.addEventListener(“input”, event => {
    console.log(inputField.value);
});
```

addEventListener - input

```
const inputField = document.querySelector(“input”);


inputField.addEventListener(“input”, event => {
    console.log(event.target.value);
});
```

addEventListener - input

To the code editor and browser

addEventListener - input

```
event.target.name
```

We can use the name to set a key dynamically

addEventListener - input

```
nameField.addEventListener(“input”, handleChange);

ageField.addEventListener(“input”, handleChange);

commentField.addEventListener(“input”, handleChange);
```

addEventListener - input

Events bubble up

addEventListener - input

To the code editor and browser

addEventListener

Any questions on click or input events?

addEventListener - submit

```
const form = document.querySelector(“form”);

form.addEventListener(“submit”, event => {

    console.log(“Submit the data!”);

});
```

addEventListener - submit

Some interactions have default behaviours:

Submitting a <form> reloads the page

Clicking an <a> tag opens the link

Clicking a checkbox toggles the checkbox

addEventListener - submit

To prevent the default behaviour

```
event.preventDefault();
```

addEventListener - submit

```
form.addEventListener(“submit”, event => {

    event.preventDefault();

    console.log(“Submit the data!”);

});
```

addEventListener - submit

To the code editor and browser

addEventListener - submit

Any questions on submit listener?

removeEventListener

addEventListener is always listening

removeEventListener takes 2 arguments

1. Event type, a string
2. The function used to add the listener

removeEventListener

```
element.addEventListener("click", event => {

    console.log("do something once");

    element.removeEventListener("click", ???);

});
```

removeEventListener

```
const handleEvent = event => {

    console.log(“do it once”);

}

element.addEventListener(“click”, handleEvent);
```

removeEventListener

```
const handleEventOnce = event => {

    console.log(“do it once”);

    event.target.removeEventListener(“click”, handleEventOnce);

}

element.addEventListener(“click”, handleEventOnce);
```

removeEventListener

To the code editor and browser

removeEventListener

Any questions on removing event listeners?

Not the only way to listen for events

on* functions

```
element.onclick = event => {
    console.log(“the element was clicked!”);
};

element.onsubmit = event => {
    console.log(“the form was submitted”);
};
```

on* functions

Pro:

- Only 1 handler at a time, no need to remove

Con:

- Only 1 handler at a time, sometimes you want multiple

on* functions

To the code editor and browser

Events

Any questions on events?