COESIONE ITALIA 21-27 PIEMONTE
Cofinanziato dall'Unione europea
REGIONE PIEMONTE

Web Developer

Programmazione - Javascript e Typescript

Docente: Shadi Lahham

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Events & Listeners

Interactivity

Shadi Lahham - Web development

Events

Why we need events

- Form validation and processing
- Interactive slideshows
- Games
- Single-page webapps
- Anything that involves user interaction

Gli eventi permettono di rendere le pagine web interattive.
Vengono usati in casi come:

- Validazione dei form e elaborazione dei dati
- Slideshow interattivi
- Giochi e applicazioni web a singola pagina
- Qualsiasi scenario che richieda interazione dell'utente.

Adding Event Listeners

creating an event listener

```
domNode.addEventListener(eventType, eventListener, useCapture);
```

example 1

```
// event listener with function written separately
function onClick(event) {
  console.log('Window clicked');
}
window.addEventListener('click', onClick);
```

example 2

```
// event listener with function written inside
window.addEventListener('click', function (event) {
  console.log('Window clicked');
});
```

References

addEventListener, type, listener, useCapture

Adding Event Listeners

index.html

```
<button id="counter">0</button>
```

main.js

```
let counterButton = document.getElementById('counter');

// event handler function
let onButtonClick = function (event) {
  counterButton.textContent = parseInt(counterButton.textContent) + 1;
  console.log('clicked element:', event.target);
};

// add click event listener to the button
counterButton.addEventListener('click', onButtonClick, false);
```

Listeners and handlers

DIFFERENZE SPIEGATE SUL WORD DI APPUNTI JS !!!

Event Listener

A function that waits for a particular event on a DOM element or the document, triggering a callback function known as the 'event handler' when the event occurs

Event Handler

The function executed when a specific event happens, also referred to as an 'event callback' responsible for responding to user or browser events such as clicks, key presses, mouse movements, form field changes, etc.

Processing input

index.html

```
<input id="myname" type="text">
<button id="button">Say My Name</button>
```

main.js

```
let button = document.getElementById('button');
let inputField = document.getElementById('myname');

// event handler
let onEvent = function (event) {
  console.log('Hi, ' + inputField.value);
  console.log('event type:', event.type);
};

// event listeners
button.addEventListener('click', onEvent); // responding to button click
inputField.addEventListener('blur', onEvent); // responding to input loss of focus
```

Event types

il tipo di evento che vogliamo ascoltare che rappresenta un occorrenza del DOM, si distinguono in diverse categorie

An Event Represents an represents an occurrence in the DOM

- mouse events - MouseEvent
  - mousedown, mouseup, click, dblclick, mousemove, mouseover, mousewheel, mouseout, contextmenu
- touch events - TouchEvent
  - touchstart, touchmove, touchend, touchcancel
- keyboard events - KeyboardEvent
  - keydown, keypress, keyup
- form events
  - focus, blur, change, submit
- window events
  - scroll, resize, hashchange, load, unload

Event types

```
// window resize handler
let onWindowResize = function (event) {
  console.log('Window resized');
};
window.addEventListener('resize', onWindowResize);

// document scroll handler
let onDocumentScroll = function (event) {
  console.log('Document scrolled');
};
document.addEventListener('scroll', onDocumentScroll);

// mousemove handling on 'myElement'
let element = document.getElementById('myElement');
let onMouseMove = function (event) {
  console.log('Mouse moved over the element');
};
element.addEventListener('mousemove', onMouseMove);
```

Event types

```
// same code with arrow functions

// window resize handler
window.addEventListener('resize', event => console.log('Window resized'));

// document scroll handler
document.addEventListener('scroll', event => console.log('Document scrolled'));

// mousemove handling on 'myElement'
let element = document.getElementById('myElement');
element.addEventListener('mousemove', event => console.log('Mouse moved over the element'));
```

Event types

```
// DOMContentLoaded is an event that fires when the initial HTML document has been completely loaded
// and parsed, without waiting for stylesheets, images, and subframes to finish loading
// It indicates that the DOM (Document Object Model) is ready to be manipulated safely

document.addEventListener('DOMContentLoaded', function () {
  // This function will be executed when the DOMContentLoaded event is fired
  // It's safe to manipulate the DOM here

  // Example: Change the text content of an element with id "example"
  let element = document.getElementById('example');
  element.textContent = 'DOM Content Loaded!';
});
```

DOMContentLoaded event | MDN

Event

An Event Represents an represents an occurrence in the DOM

When an event happens, the browser generates an event object with details like event type (click, keydown, mousemove), the target element, and relevant data (mouse coordinates, keyboard key)

Important methods and properties

```
event.stopPropagation();
event.target;
event.currentTarget;
event.preventDefault();
event.type;
event.target;
```

The .parentElement property of a Node can sometimes be useful in event handling

Event target

index.html

```
<ul id="list">
  <li>item 1 <button>highlight</button></li>
  <li>item 2 <button>highlight</button></li>
</ul>
```

style.css

```
.highlighted {background-color: orange;}
```

main.js

```
document.getElementById('list').addEventListener('click', event => {
  const target = event.target;
  // if  clicked element is a button
  if (target.tagName === 'BUTTON') {
    const parent = target.parentElement;
    parent.classList.toggle('highlighted'); // mark as highlighted
  }
});
```

Event object

```
// log coordinates when mouse moves
document.addEventListener('mousemove', event => {
  console.log('Mouse coordinates:', {
    x: event.clientX,
    y: event.clientY
  });
});

// log key press
document.addEventListener('keydown', event => console.log('Key pressed:', event.key));

// log event timestamp
document.addEventListener('click', event => console.log('Event timestamp:', event.timeStamp));
```

Bubbling and capturing

Gli eventi propagano attraverso il DOM in due fasi:

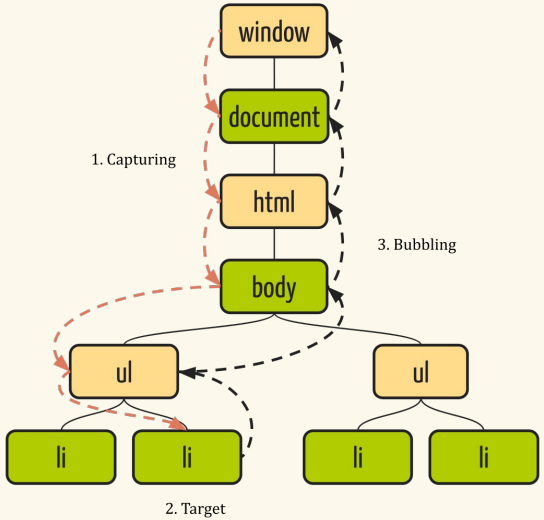
- CAPTURING: L'evento si propaga dall'antenato esterno fino all'elemento target.

- BUBBLING: Dopo che l'evento ha raggiunto il target, si propaga in senso opposto, risalendo gli antenati.

Bubbling and capturing

Phases of event propagation

1. Capturing
2. Target
3. Bubbling



Bubbling and capturing

Bubbling

The event listener is added with the default behavior of bubbling, in which the event starts from the target element and propagates up through its ancestor elements

```
node.addEventListener('click', handler); // uses bubbling
```

Capturing

The event listener is set to use capturing by setting the useCapture parameter to true, which involves the event being initially captured by the outermost ancestor element and then propagating down to the target element

```
node.addEventListener('click', handler, true); // uses capturing
```

Bubbling and capturing

Events in JavaScript propagate through the DOM tree, traversing from the target element to its ancestors

The concepts covered in the following two links are fundamental to this unit's material so please study them in detail

- Bubbling and capturing
- Event Bubbling in JavaScript? Event Propagation Explained

Removing an event listener

si può rimuovere un evento specifico

```
node.addEventListener('click', handler);
node.removeEventListener('click', handler);      // succeeds

node.addEventListener('click', handler, true);
node.removeEventListener('click', handler, false);     // fails
node.removeEventListener('click', handler, true);      // succeeds
```

When a listener is registered both with and without the capture flag, each one must be removed separately, as removing a capturing listener does not affect a non-capturing version of the same listener, and vice versa

EventTarget.removeEventListener()

Get phase example

```
<div id="container">
  <ul>
    <li>item1</li>
    <li id="special">item2</li>
    <li>item3</li>
  </ul>
</div>

let container = document.getElementById('container');
let special = document.getElementById('special');

special.addEventListener('click', function(event) {
  console.log('Special', getLabel(event.eventPhase));
});
```

Get phase example

```
// capturing
container.addEventListener('click', function(event) {
  console.log('Container', getLabel(event.eventPhase));
},true);

// bubbling
container.addEventListener('click', function(event) {
  console.log('Container', getLabel(event.eventPhase));
});

// get phase label
function getLabel(phase) {
  return phase === 3 ? 'bubbling' : phase === 2 ? 'on target' : 'capturing';
}
```

Animation

The window object

When you run Javascript in the browser, you can access the window object which has many useful properties and methods

Examples

```
window.location.href;
window.navigator.userAgent;
window.scrollTo(10, 50);
window.console.log('Hello world!');
```

Window has may other useful properties and methods
The window object is the assumed global object on a page

```
window.console.log('hi');
// is the same as
console.log('hi');
```

Animating with Javascript

Animations should be done using CSS transitions and CSS animations and not with Javascript

Here is an example just to show that it's possible, but not at all recommended

Using the timing functions

```
window.setTimeout(callbackFunction, delayMilliseconds);
window.setInterval(callbackFunction, delayMilliseconds);
```

```
// just an example
let makeImageBigger = function () {
  let img = document.getElementsByTagName('img')[0];
  img.setAttribute('width', img.width + 10);
};
window.setInterval(makeImageBigger, 1000);
```

Animating CSS styles

You can animate CSS styles to change size, transparency, position, color, etc
Again, you shouldn't do this with Javascript

```
// just an example
let fadeAway = function () {
  img.style.opacity = img.style.opacity - 0.1;
};

let img = document.getElementsByTagName('img')[0];
img.style.opacity = 1.0;
window.setInterval(fadeAway, 1000);
```

Animating CSS styles

```
// what happens to kitty?

let watchKittyFall = function () {
  let oldTop = parseInt(img.style.top);
  let newTop = oldTop + 10;
  img.style.top = newTop + 'px';
};

let img = document.getElementsByTagName('img')[0];
img.style.position = 'absolute';
img.style.top = '0px';
window.setInterval(watchKittyFall, 1000);
```

Stopping Animations

```
// to stop an animation store the timer into a variable and clear it
window.clearTimeout(timer);
window.clearInterval(timer);

// example
let fadeAway = function () {
  img.style.opacity = img.style.opacity - 0.1;
  if (img.style.opacity < 0.5) {
    window.clearInterval(fadeTimer);
  }
};

let img = document.getElementsByTagName('img')[0];
img.style.opacity = 1.0;
let fadeTimer = window.setInterval(fadeAway, 100);
```

Your turn

1.Story

Start with the following HTML:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Story</title>
</head>
<body>
  <h1>Story</h1>
  <ul>
    <li>Noun: <input type="text" id="noun">
    <li>Adjective: <input type="text" id="adjective">
    <li>Someone's Name: <input type="text" id="person">
  </ul>
  <button id="gen-button">Lib it!</button>
  <div id="story"></div>
</body>
</html>
```

Continues on next page >>>

1.Story

- Add an event listener to the button so that it calls a makeStory function when clicked.
- In the makeStory function, retrieve the current values of the form input elements, make a story from them, and output that in the story div (like "Joseph really likes pink cucumbers.")

2.Calculate

Start with the following HTML:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Calculator</title>
</head>
<body>
  <label>Square this number:
    <input type="number" id="square-input" size="2">
  </label>
  <button id="square-button">Calculate</button>
  <!--other inputs here -->

  <div id="solution"></div>
</body>
</html>
```

Continues on next page >>>

2.Calculate

- Add inputs for half number, percentage and circle area
- Use the functions from the previous calculator exercises
- For each operation, create an event listener for the button, and when it's clicked, find the value of the appropriate input and show the result of the calculation in the solution div
- Afterwards, change the code so that you respond to key presses so that the user doesn't have to click the button

3.Catwalk

Start with the following HTML:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Cat Walk</title>
</head>
<body>

  <img style="position:absolute;" src="http://www.anniemation.com/clip_art/images/cat-walk.gif">

</body>
</html>
```

Continues on next page >>>

3.Catwalk

- The cat should start from the left side of the screen
- Write a function ‘catWalk()’ that moves the cat 10 pixels to the right
- Make the cat move across the screen by calling that function every 50ms
- Write different versions of the function to handle the following variants:
  - Variant 1: When the cat reaches the right side of the screen it should restart from the left
  - Variant 2: When the cat reaches the right side of the screen, it should move backwards. When it reaches the left it should move forwards
  - Variant 3: When the cat reaches the middle of the screen, replace the img with a different cat image. Keep it in the middle for 10 seconds, and then replace the img with the original image and have it continue the walk as in variant 2

Bonus

4.Enhanced Catwalk

- Start with the code from the previous ‘Catwalk’ exercise
- Add 4 buttons at the top of the page: ‘start’, ‘faster’, ‘slower’ and ‘stop’
- Add an area to display info
- When the start button is clicked the cat should start moving across the screen
- The cat should stop moving when the stop button is clicked
- The cat moves faster when the faster button is clicked and slower when the slower button is clicked
- Show the current speed on screen in the info area
- Disable the start/stop/faster/slower buttons at the appropriate times
    - e.g. the user shouldn't be able to click "stop" if the cat isn't currently moving

5.Enhanced Arrivals

- Start with the ‘Arrivals’ exercise from a previous lesson
- Add the following features:
  - When the user clicks a row, it should expand to show more information about the flight
  - When the user clicks an open row it should close again
  - If the user clicks a row, any other open rows should close
    - Like in this example of an accordion
  - Add a ‘Departures’ section with departing flights
  - The user should be able to switch between Arrivals and Departures with a fade-in/fade-out animation

6.Enhanced Word Guesser

- Begin with the 'Word Guesser' exercise from a previous lesson
- Incorporate the following enhancements
  - Implement an interface to allow users to input letters via an HTML <input> element
    - Integrate additional necessary HTML elements
    - Validate user input to ensure it consists of a single character and is of the correct type
  - Enhance the visual presentation of game results directly on the page
    - This can include the utilization of images or other graphics to enrich the experience
    - Display previous guesses, the remaining number of guesses, and other pertinent information to the user
  - Add additional features that would augment the game's appeal and interactivity

Bonus Challenge: Develop a 2-player version of the game to further engage users in a competitive gameplay experience

References

Introduction to events
Event - Web APIs | MDN

Bubbling and capturing
Event Bubbling in JavaScript? Event Propagation Explained

Event.stopPropagation()
Event.target
Event.currentTarget
EventTarget.removeEventListener()