EPICODE

JavaScript 2

Giorno 3

EPICODE

DOM Events, Form Validation, BOM

Una delle potenzialità maggiori di JS è quella di poter rispondere a degli eventi richiamati sugli elementi HTML.

Per esempio, il click su un bottone scatena un certo tipo di evento programmabile.

Tutti gli eventi del DOM derivano dall'oggetto Event.

Tipologie comuni di eventi

- Eventi del mouse: click, hover...
- Eventi della tastiera
- Caricamento di una pagina
- Caricamento di una immagine
- Eventi sul campo dell'input
- Invio di un form HTML

Esempio

Il metodo onclick può essere utilizzato per invocare una certa funzione che gestisce un evento.

```
<button onclick="funzione()">Clicca</button>
```

Mostra un testo nell'elemento selezionato.

La catena degli eventi

addEventListener

Scopo principale del metodo è assegnare ad un elemento selezionato un certo evento.

```
elemento.addEventListener(evento, funzione);
```

Possiamo però anche attribuire più eventi e funzioni al medesimo elemento.

```
document.getElementById(“nomeID”)
.addEventListener(“click”, nomeFunzione);
```

```
document.getElementTagName(“button”)
.addEventListener(“click”, nomeFunzione);

document.getElementTagName(“button”)
.addEventListener(“mouseover”, nomeFunzione1);

document.getElementTagName(“button”)
.addEventListener(“mouseleave”, nomeFunzione2);
```

I nodi possono essere agevolmente manipolati aggiungendone di nuovi o rimuovendo gli esistenti.

Una collezione (lista) di elementi può essere Generata usando il metodo getElementsByTagName che richiama ovviamente tutti gli elementi contrassegnati dal tag.

```
//Aggiungere un nodo
document.createElement("h1");
document.createTextNode(“Un nuovo titolo”);

//Rimuovere un nodo
elemento.remove();
elemento.childRemove(elemento selezionato);
```

```
document.getElementsByTagName(“p”);
```

Il metodo genera un oggetto che possiamo trattare in maniera analoga ad un array.

In questo caso abbiamo un array di tag paragrafo.

EPICODE

Form e validazione

Manipolazione degli elementi del form

JavaScript utilizza un oggetto Form per metodi e proprietà di gestione di elementi del form.

È possibile accedere ai form del documento direttamente con la HTMLCollection forms:

```
document.forms[0]; // primo form della pagina
document.forms[1]; // secondo form della pagina
```

Per ogni form è anche possibile esplorarne gli elementi tramite la proprietà elements:

```
document.forms[0].elements[0]; // primo elemento
```

La proprietà length ci riporta il numero di form del documento oppure degli elementi in un form.

All'oggetto Form possiamo attribuire due metodi principali: reset() e submit().

L'operazione di submit, lavorando con il moderno JS dev'essere obbligatoriamente eseguita inserendo l'esecuzione del metodo preventDefault() sull'evento scatenante, in modo da fermare il comportamento predefinito del browser.

Validazione dei form

La validazione dei campi di input di un form richiede sempre massima attenzione.

Tipologia di validazione:

- Campi vuoti
- Controllo di inserimento numerico o testuale
- Inserimento di sequenze numeriche di una determinata lunghezza
- Inserimento di sequenze con controllo di una espressione regolare come nelle password o nelle mail.

La validazione lato client controlla la correttezza dell'input prima che il dato sia inviato al server, quindi il controllo è a livello del browser.

API di validazione

Oltre le strutture di controllo condizionale con le quali possiamo validare un campo, JavaScript sfrutta le potenzialità di API dedicata con il metodi:

checkValidity() ⇒ controllo della validità dei dati

Per esempio si può controllare che un numero inserito in un input sia all'interno di un certo range.

Fra le proprietà troviamo:

validationMessage

che ritorna un messaggio quando la condizione di validità è falsa

validity

con la quale è possibile settare diverse opzioni di controllo

Il controllo di validità fa largo uso degli attributi dei form propri di HTML5 come min e max

Esempio:
<input type="number" min="50" max="100">

Altri attributi utilizzabili sono minlength e maxlength, che controllano la lunghezza del contenuto inserito.

EPICODE

BOM e JavaScript

Con JavaScript possiamo manipolare e gestire dimensioni ed eventi legati alla finestra del browser.

Ecco i metodi utilizzabili con l'oggetto:

- window.open()
- window.close()
- window.moveTo()
- window.resizeTo()
- window.innerHeight
- window.innerWidth

Con JavaScript possiamo utilizzare 3 tipi di pop up gestiti dall' oggetto window:

- Pop up di alert ⇒ window.alert();
- Pop up di conferma ⇒ window.confirm();
- Pop up di prompt ⇒ window.prompt();

I metodi possono essere scritti omettendo window:

- alert('valore');
- confirm('valore');
- prompt('valore' , 'testo di default');

Con JavaScript possiamo manipolare e gestire la storia legata agli eventi di navigazione.

L'oggetto è:

window.history

con i metodi:

```
history.back();
history.forward();
```

che replicano i relativi tasti del browser.

Con l'oggetto Navigator possiamo ricavare alcune informazioni di navigazione dell'utente relative al browser utilizzato.

L'oggetto window.navigator ha una discreta lista di metodi fra cui i principali:

- navigator.appName
- navigator.appCodeName
- navigator.platform ⇒ sistema operativo
- navigator.product
- navigator.appVersion ⇒ versione del browser
- navigator.userAgent
- navigator.language
- navigator.online

L'oggetto window usa due metodi per il settaggio dei tempi di azione della finestra del browser:

setTimeout(*funzione*, 1000);
esegue una funzione dopo un certo tempo

setInterval(*funzione*, 1000);
ripete ciclicamente secondo il tempo impostato una funzione

Gli eventi dei due metodi possono essere cancellati con:

clearTimeout();

clearInterval();

EPICODE

GRAZIE

EPICODE