GABRIELE ROMANATO

Creare slideshow JavaScript: tutorial completo

In questo articolo tratteremo le basi degli slideshow JavaScript. Descriveremo i blocchi fondamentali che costituiscono uno slideshow JavaScript (HTML, CSS, JavaScript) e le tecniche usate per gestire gli slideshow. Dobbiamo concentrarci sulle basi se vogliamo poi sviluppare slideshow più complessi e avanzati.

Il codice JavaScript verrà sviluppato sia in semplice JavaScript che con jQuery. Il motivo è semplice: oggi possiamo combinare JavaScript con le animazioni e transizioni CSS ma allo stesso tempo abbiamo bisogno di implementare anche soluzioni che funzionino in tutti i browser.

Verrà usato un approccio OOP negli esempi. Se non avete familiarità con questo modo di scrivere il codice JavaScript potete iniziare da questo thread su Stack Overflow.

Sommario

        - Esempi
    - Elementi multimediali: video
        - Esempi
    - Slideshow automatici
        - Esempi
    - Navigazione da tastiera
        - Esempi
    - Callback
        - Esempi
    - Usare le API
        - Esempi
- Conclusione
- Esempi completi

La struttura HTML

Dobbiamo tenere presente che la marcatura deve mantenere una struttura semantica anche in assenza di CSS e JavaScript. Per far questo dobbiamo conoscere quali componenti faranno parte della nostra struttura. Di solito questi componenti sono:

1. Un contenitore esterno.
2. Un contenitore interno (facoltativo).
3. Le slide.
4. I link della paginazione (facoltativi).
5. Due bottoni per andare avanti ed indietro nella sequenza delle slide (facoltativi).

I componenti 2, 4 e 5 sono facoltativi per i seguenti motivi:

- Le slide possono essere racchiuse da un singolo elemento, di solito quando l'effetto usato è di assolvenza/dissolvenza.
- I link della paginazione e i bottoni possono essere omessi quando gli slideshow sono automatici.

Ecco una possibile struttura HTML:

```
<div class="slider" id="main-slider"><!-- contenitore esterno
-->
        <div class="slider-wrapper"><!-- contenitore interno
-->
                <div class="slide">...</div><!-- slide -->
                <div class="slide">...</div>
                <div class="slide">...</div>
        </div>
        <div class="slider-nav"><!-- "Precedente" e
"Successivo" -->
                <button type="button" class="slider-
previous">Precedente</button>
                <button type="button" class="slider-
next">Successivo</button>
        </div>
</div>
```

Si raccomanda di usare le classi sugli elementi degli slideshow perché potrebbero esserci più slideshow nella stessa pagina. Per identificarli univocamente possiamo usare un ID sul contenitore esterno.

Usiamo elementi `button` perché i link in questo caso non punterebbero a nessuna risorsa effettiva: potete consultare You can't create a button di *Nicholas Zakas* per approfondire i motivi di questa scelta.

Se le slide sono immagini, la struttura può essere modificata come segue:

```
<div class="slider" id="main-slider"><!-- contenitore esterno
-->
        <div class="slider-wrapper"><!-- contenitore interno
-->
                <img src="image1.jpg" alt="Descrizione"
class="slide" /><!-- slide -->
                <img src="image2.jpg" alt="Descrizione"
```

```
class="slide" />
                <img src="image3.jpg" alt="Descrizione"
class="slide" />
        </div>
        <div class="slider-nav"><!-- "Precedente" e
"Successivo" -->
                <button type="button" class="slider-
previous">Precedente</button>
                <button type="button" class="slider-
next">Successivo</button>
        </div>
</div>
```

Bisogna sempre usare attributi `alt` sulle immagini: non dimentichiamo mai l'accessibilità della nostra implementazione. Per usare i link della paginazione possiamo modificare il codice come segue:

```
<div class="slider" id="main-slider"><!-- contenitore esterno
-->
        <div class="slider-wrapper"><!-- contenitore interno
-->
                <div class="slide" id="slide-1">...</div><!--
slide -->
                <div class="slide" id="slide-2">...</div>
                <div class="slide" id="slide-3">...</div>
        </div>
        <div class="slider-nav"><!-- link della paginazione -
->
                <a href="#slide-1">1</a>
                <a href="#slide-2">2</a>
                <a href="#slide-3">3</a>
        </div>
</div>
```

Abbiamo fatto puntare ciascun link ad una slide grazie all'ancora impostata sull'attributo `href`. Ricordate che il nostro slideshow deve essere

accessibile anche senza JavaScript.

Altri slideshow combinano i bottoni di controllo con i link della paginazione:

```
<div class="slider" id="main-slider"><!-- contenitore esterno
-->
        <div class="slider-wrapper"><!-- contenitore interno
-->
            <!-- slide -->
                <div class="slide" id="slide-1"</div>
                <div class="slide" id="slide-2"</div>
                <div class="slide" id="slide-3"></div>
        </div>
        <!-- "Precedente" e "Successivo" -->
        <div class="slider-nav">
                <button type="button" class="slider-
previous">Precedente</button>
                <button type="button" class="slider-
next">Successivo</button>
        </div>
        <!-- link della paginazione -->
        <div class="slider-pagination">
                <a href="#slide-1">1</a>
                <a href="#slide-2">2</a>
                <a href="#slide-3">3</a>
        </div>
</div>
```

Usare le liste

Se si considerano le slide come una serie di elementi, allora possiamo modificare la nostra struttura HTML come segue:

```
<ul class="slider-wrapper"><!-- contenitore interno -->
                <li class="slide" id="slide-1">...</li><!--
slide -->
                <li class="slide" id="slide-2">...</li>
```

```
                <li class="slide" id="slide-3">...</li>
</ul>
```

Se le slide sono in un ordine preciso (ad esempio una presentazione), potete anche usare liste ordinate (`<ol>`).

Il codice CSS

Consideriamo la seguente struttura HTML:

```
<div class="slider" id="main-slider"><!-- contenitore esterno
-->
        <div class="slider-wrapper"><!-- contenitore interno
-->
                <img src="image1.jpg" alt="Descrizione"
class="slide" /><!-- slide -->
                <img src="image2.jpg" alt="Descrizione"
class="slide" />
                <img src="image3.jpg" alt="Descrizione"
class="slide" />
        </div>
        <div class="slider-nav"><!-- "Precedente" e
"Successivo"  -->
                <button type="button" class="slider-
previous">Precedente</button>
                <button type="button" class="slider-
next">Successivo</button>
        </div>
</div>
```

Il movimento avrà luogo da destra verso sinistra. Ciò significa che il contenitore esterno avrà una data larghezza mentre il contenitore interno sarà più largo per contenere tutte le slide. Solo la prima slide sarà visibile inizialmente: per far ciò usiamo la proprietà CSS `overflow`.

```
.slider {
        width: 1024px;
        overflow: hidden;
}

.slider-wrapper {
        width: 9999px;
        height: 683px;
        position: relative;
        transition: left 500ms linear;
}
```

Il contenitore interno ha i seguenti stili importanti:

- Una larghezza tale da contenere tutte le slide.
- Un'altezza stabilita per contenere le slide che saranno flottate.
- Il posizionamento relativo per creare il movimento da destra verso sinistra.
- Una transizione CSS sulla proprietà `left` che creerà il movimento. Potete usare anche una traslazione con le trasformazioni CSS. Se usate jQuery potete anche non usare questa tecnica CSS.

Le slide sono flottate e posizionate in modo relativo. Il posizionamento relativo in questo caso serve a JavaScript per calcolare esattamente l'offset di ciascuna slide.

```
.slide {
        float: left;
        position: relative;
        width: 1024px;
        height: 683px;
}
```

La larghezza del contenitore interno verrà sovrascritta da JavaScript: 9999 pixels può anche non bastare se lo slideshow è a pieno schermo e ci sono molte slide.

Per applicare gli stili ai bottoni "Precedente" e "Successivo" dobbiamo prima resettare i loro stili predefiniti:

```
.slider-nav {
	height: 40px;
	width: 100%;
	margin-top: 1.5em;
}

.slider-nav button {
	border: none;
	display: block;
	width: 40px;
	height: 40px;
	cursor: pointer;
	text-indent: -9999em;
	background-color: transparent;
	background-repeat: no-repeat;
}

.slider-nav button.slider-previous {
	float: left;
	background-image: url(previous.png);
}

.slider-nav button.slider-next {
	float: right;
	background-image: url(next.png);
}
```

Per i link della paginazione possiamo invece usare i seguenti stili:

```
.slider-nav {
```

```
        text-align: center;
        margin-top: 1.5em;
}

.slider-nav a {
        display: inline-block;
        text-decoration: none;
        border: 1px solid #ddd;
        color: #444;
        width: 2em;
        height: 2em;
        line-height: 2;
        text-align: center;
}
.slider-nav a.current {
        border-color: #000;
        color: #000;
        font-weight: bold;
}
```

La classe `current` verrà applicata dinamicamente da JavaScript quando viene selezionata una slide.

Questo tipo di layout è comune quando l'effetto voluto è uno scorrimento. Se invece vogliamo un effetto di assolvenza e dissolvenza dobbiamo usare il posizionamento contestuale per creare uno stack di slide:

```
.slider {
        width: 1024px;
        margin: 2em auto;

}

.slider-wrapper {
        width: 100%; /* La larghezza ora è uguale a quella
del contenitore esterno */
        height: 683px;
        position: relative; /* Crea il contesto per il
posizionamento assoluto */
```

```
}

.slide {
        position: absolute; /* Tutte le slide sono
posizionate in modo assoluto */
        width: 100%;
        height: 100%;
        opacity: 0; /* Le slide sono nascoste */
        transition: opacity 500ms linear; /* Transizione su
opacity */
}

/* Solo la prima slide è visibile inizialmente */

.slider-wrapper > .slide:first-child {
        opacity: 1;
}
```

Usiamo `opacity` perchè `display: none` sopprime la lettura del contenuto negli screen reader (consultate CSS in Action: Invisible Content Just for Screen Reader Users di *WebAIM* per maggiori dettagli).

Il posizionamento contestuale da noi creato segue l'ordine imposto dallo stacking: l'ultima slide sarà la prima ad essere visualizzata. Dato che non vogliamo questo effetto, nascondiamo tutte le slide tranne la prima.

JavaScript cambierà il valore di `opacity` sulla slide corrente e resetterà tale valore su 0 su tutte le altre. Se usate jQuery potete anche usare direttamente il metodo `.animate()`.

Compatibilità con IE9

IE9 non supporta le transizioni CSS. Se vogliamo mantenere la compatibilità con questo browser dovremmo usare jQuery.

Per una panoramica sulle possibili soluzioni, consultate questo thread su Stack Overflow.

Il codice JavaScript

Slideshow senza paginazione

Gli slideshow senza paginazione usano solo i bottoni "Precedente" e "Successivo". Questi bottoni servono per incrementare e decrementare un contatore interno. Il contatore non è altro che un indice numerico usato per selezionare le slide. Inizialmente è impostato su 0 e il suo funzionamento è identico a quello degli indici degli array.

Così cliccando una prima volta sul pulsante "Successivo" il contatore avrà il valore di 1 e selezioneremo la seconda slide (la numerazione negli array comincia da 0). Quindi cliccando sul pulsante "Precedente" il contatore avrà valore 0 e selezioneremo di nuovo la prima slide. E così via.

Con jQuery possiamo usare il metodo `.eq()` con il nostro contatore, ma in JavaScript l'approccio è diverso:

```
function Slideshow( element ) {
        this.el = document.querySelector( element );
        this.init();
}

Slideshow.prototype = {
        init: function() {
                this.slides = this.el.querySelectorAll(
".slide" ); // Una NodeList è simile ad un array
                //...
        },
        _slideTo: function( pointer ) {
                var currentSlide = this.slides[pointer]; //
Slide corrente
                //...
        }
};
```

Una `NodeList` usa gli indici come un vero array. Un altro modo è quello di usare i selettori CSS3:

```
Slideshow.prototype = {
        init: function() {
                //...
        },
        _slideTo: function( pointer ) {

        var n = pointer + 1; // :nth-child() conta a partire
da 1, non da 0
                var currentSlide = this.el.querySelector(
".slide:nth-child(" + n + ")" ); // Slide corrente
                //...
        }
};
```

Il selettore `:nth-child()` conta a partire da 1 e non da 0, quindi dobbiamo incrementare il contatore di 1 o più semplicemente impostare il suo valore iniziale su 1 per selezionare le slide.

Una volta selezionata una slide la usiamo per far scorrere il contenitore interno da destra a sinistra. In jQuery usiamo il metodo `.animate()` con un valore negativo per la proprietà `left` del contenitore interno. Il valore negativo è ricavato dall'offset della slide selezionata:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
                options = $.extend({
                        wrapper: ".slider-wrapper",
                        slides: ".slide",
                        //...
                        speed: 500,
                        easing: "linear"
                }, options);
```

```
                var slideTo = function( slide, element ) {
                        var $currentSlide = $(
options.slides, element ).eq( slide );

                        $( options.wrapper, element ).
                        animate({
                                left: -
$currentSlide.position().left
                        }, options.speed, options.easing );

                };

        //...
        };

})( jQuery );
```

In JavaScript non esiste un metodo nativo `.animate()`, quindi dobbiamo usare le transizioni CSS:

```
.slider-wrapper {
        position: relative; // Richiesto
        transition: left 500ms linear;
}
```

Quindi modifichiamo il valore della proprietà `left` usando l'oggetto `style`:

```
function Slideshow( element ) {
        this.el = document.querySelector( element );
        this.init();
}

Slideshow.prototype = {
        init: function() {
            this.wrapper = this.el.querySelector( ".slider-
wrapper" );
```

```
                this.slides = this.el.querySelectorAll(
".slide" );
                //...
        },
        _slideTo: function( pointer ) {
                var currentSlide = this.slides[pointer];
                this.wrapper.style.left = "-" +
currentSlide.offsetLeft + "px";
        }
};
```

Ora dobbiamo legare un evento `click` a ciascun pulsante. In jQuery usiamo il metodo `.on()` mentre in JavaScript usiamo il metodo `addEventListener()`.

Dobbiamo anche verificare che il nostro contatore non abbia raggiunto un valore specifico, ossia 0 per il pulsante "Precedente" e il numero totale di slide per il pulsante "Successivo". In entrambi i casi i pulsanti devono essere mostrati o nascosti ed il contatore resettato sul valore corretto.

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
                options = $.extend({
                        wrapper: ".slider-wrapper",
                        slides: ".slide",
                        previous: ".slider-previous",
                        next: ".slider-next",
                        //...
                        speed: 500,
                        easing: "linear"
                }, options);

                var slideTo = function( slide, element ) {
                        var $currentSlide = $(
options.slides, element ).eq( slide );

                        $( options.wrapper, element ).
                        animate({
```

```
                                left: -
$currentSlide.position().left
                        }, options.speed, options.easing );

                };

        return this.each(function() {
                        var $element = $( this ),
                                $previous = $(
options.previous, $element ),
                                $next = $( options.next,
$element ),
                                index = 0,
                                total = $( options.slides
).length;

                        $next.on( "click", function() {
                                index++;
                                $previous.show();

                                if( index == total - 1 ) { //
Numero totale
                                        index = total - 1;
                                        $next.hide();
                                }

                                slideTo( index, $element );

                        });

                        $previous.on( "click", function() {
                                index--;
                                $next.show();

                                if( index == 0 ) {
                                        index = 0;
                                        $previous.hide();
                                }

                                slideTo( index, $element );

                        });
```

```
                });
        };

})( jQuery );
```

In JavaScript invece il codice sarà come segue:

```
function Slideshow( element ) {
        this.el = document.querySelector( element );
        this.init();
}

Slideshow.prototype = {
        init: function() {
            this.wrapper = this.el.querySelector( ".slider-
wrapper" );
                this.slides = this.el.querySelectorAll(
".slide" );
                this.previous = this.el.querySelector(
".slider-previous" );
                this.next = this.el.querySelector( ".slider-
next" );
                this.index = 0;
                this.total = this.slides.length;

                this.actions();
        },
        _slideTo: function( pointer ) {
                var currentSlide = this.slides[pointer];
                this.wrapper.style.left = "-" +
currentSlide.offsetLeft + "px";
        },
        actions: function() {
                var self = this;
                self.next.addEventListener( "click",
function() {
                        self.index++;
                        self.previous.style.display =
```

```
"block";

                        if( self.index == self.total - 1 ) {
                                self.index = self.total - 1;
                                self.next.style.display =
"none";
                        }

                        self._slideTo( self.index );

                }, false);

                self.previous.addEventListener( "click",
function() {
                        self.index--;
                        self.next.style.display = "block";

                        if( self.index == 0 ) {
                                self.index = 0;
                                self.previous.style.display =
"none";
                        }

                        self._slideTo( self.index );

                }, false);
        }

};
```

Esempi

- jQuery: full width slideshow
- Creating a JavaScript slideshow: using plain JavaScript

Slideshow con paginazione

Negli slideshow con paginazione ciascun link della paginazione corrisponde ad una singola slide. Non c'è più bisogno di avere un contatore interno

poiché ciascun link contiene un'ancora che punta all'ID della slide corrispondente.

Le animazioni non cambiano. In jQuery abbiamo il seguente codice:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
                options = $.extend({
                        wrapper: ".slider-wrapper",
                        slides: ".slide",
                        nav: ".slider-nav",
                        speed: 500,
                        easing: "linear"
                }, options);

                var slideTo = function( slide, element ) {
                        var $currentSlide = $(
options.slides, element ).eq( slide );

                        $( options.wrapper, element ).
                        animate({
                                left: -
$currentSlide.position().left
                        }, options.speed, options.easing );

                };

        return this.each(function() {
                        var $element = $( this ),
                                $navigationLinks = $( "a",
options.nav );

                                $navigationLinks.on( "click",
function( e ) {
                                        e.preventDefault();
                                        var $a = $( this ),
                                                $slide = $(
$a.attr( "href" ) );

                                                slideTo(
$slide, $element );
```

```
                                                $a.addClass(
"current" ).siblings().
                                                removeClass(
"current" );

                                });


                });
        };

})( jQuery );
```

In JavaScript avremo:

```
function Slider( element ) {
        this.el = document.querySelector( element );
        this.init();
}
Slider.prototype = {
        init: function() {
                this.links = this.el.querySelectorAll(
"#slider-nav a" );
                this.wrapper = this.el.querySelector(
"#slider-wrapper" );
                this.navigate();
        },
        navigate: function() {
                for ( var i = 0; i < this.links.length; ++i )
{
                        var link = this.links[i];
                        this.slide( link );
                }
        },
        slide: function( element ) {
                var self = this;
                element.addEventListener( "click", function(
e ) {
                        e.preventDefault();
```

```
                        var a = this;
                        self.setCurrentLink( a );
                        var index = parseInt( a.getAttribute(
"data-slide" ), 10 ) + 1;
                        var currentSlide =
self.el.querySelector( ".slide:nth-child(" + index + ")" );
                        self.wrapper.style.left = "-" +
currentSlide.offsetLeft + "px";
                },
                false);
        },
        setCurrentLink: function(link) {
                var parent = link.parentNode;
                var a = parent.querySelectorAll( "a" );
                link.className = "current";
                for ( var j = 0; j < a.length; ++j ) {
                        var cur = a[j];
                        if ( cur !== link ) {
                                cur.className = "";
                        }
                }
        }
};
```

Da IE10 in poi potete manipolare le classi CSS con l'oggetto classList:

```
link.classList.add( "current" );
```

Da IE11 in poi potete gestire gli attributi di dati con la proprietà dataset:

```
var index = parseInt( a.dataset.slide, 10 ) + 1;
```

Esempi

- jQuery: full width slideshow with pagination
- JavaScript: slideshow with pagination

Slideshow con paginazione e pulsanti di controllo

Questo tipo di slideshow è particolare: dobbiamo combinare il contatore interno con le ancore dei link della paginazione. In altre parole quando clicchiamo sul terzo link della paginazione il contatore dovrebbe essere impostato su 2 e quando clicchiamo sul pulsante "Precedente" il contatore dovrebbe avere come valore 1. Quindi si tratta di sincronizzare i link della paginazione con i pulsanti "Precedente" e "Successivo".

Possiamo usare l'indice numerico degli elementi del DOM: quando clicchiamo sul terzo link della paginazione l'indice sarà 2 e lo useremo per impostare il nostro contatore. Inoltre su ogni link della paginazione dovremo effettuare gli stessi controlli che abbiamo implementato per i pulsanti "Precedente" e "Successivo".

In jQuery avremo questo codice:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
                options = $.extend({
                        //...
                        pagination: ".slider-pagination",
                        //...

                }, options);

                $.fn.slideshow.index = 0;

                return this.each(function() {
                        var $element = $( this ),
                            //...
                            $pagination = $(
options.pagination, $element ),
                            $paginationLinks = $( "a",
$pagination ),
```

```
                        //...

                    $paginationLinks.on( "click", function( e
) {
                                e.preventDefault();
                                var $a = $( this ),
                                        elemIndex =
$a.index(); // Indice numerico del DOM
                                        $.fn.slideshow.index
= elemIndex;

                                        if(
$.fn.slideshow.index > 0 ) {

$previous.show();

                                        } else {

$previous.hide();
                                        }

                                        if(
$.fn.slideshow.index == total - 1 ) {

$.fn.slideshow.index = total - 1;
                                                $next.hide();
                                        } else {
                                                $next.show();
                                        }



                                        slideTo(
$.fn.slideshow.index, $element );
                                        $a.addClass(
"current" ).

siblings().removeClass( "current" );

                        });

            });
```

```
        };
        //...
})( jQuery );
```

Il nostro contatore è stato dichiarato come proprietà dell'oggetto `slideshow`: in questo modo evitiamo di incorrere nei problemi di scope creati indirettamente dai metodi di callback di jQuery. Così il contatore è accessibile sia dentro che fuori dal namespace del nostro plugin.

Per ottenere l'indice di ciascun link della paginazione usiamo il metodo .index() di jQuery. Quindi impostiamo il nostro contatore sul valore ottenuto.

In JavaScript non c'è un metodo `.index()`, quindi la soluzione più rapida è usare gli attributi di dati con un loop `for`:

```
(function() {

        function Slideshow( element ) {
                this.el = document.querySelector( element );
                this.init();
        }

        Slideshow.prototype = {
                init: function() {
                        this.wrapper = this.el.querySelector(
".slider-wrapper" );
                        this.slides =
this.el.querySelectorAll( ".slide" );
                        this.previous =
this.el.querySelector( ".slider-previous" );
                        this.next = this.el.querySelector(
".slider-next" );
                        this.navigationLinks =
this.el.querySelectorAll( ".slider-pagination a" );
                        this.index = 0;
                        this.total = this.slides.length;

                        this.setup();
```

```
                        this.actions();
                },
        //...
            setup: function() {
                    var self = this;
                    //...
                    for( var k = 0; k <
self.navigationLinks.length; ++k ) {
                                var pagLink =
self.navigationLinks[k];
                                pagLink.setAttribute( "data-
index", k );
                                // Oppure:
pagLink.dataset.index = k;
                        }
            },
            //...
    };
})();
```

Quindi possiamo scrivere:

```
actions: function() {

        var self = this;

        //...

        for( var i = 0; i < self.navigationLinks.length; ++i
) {
                var a = self.navigationLinks[i];

                a.addEventListener( "click", function( e ) {
                        e.preventDefault();
                        var n = parseInt( this.getAttribute(
"data-index" ), 10 );
                        // Oppure: var n = parseInt(
this.dataset.index, 10 );
```

```
                        self.index = n;

                        if( self.index == 0 ) {
                                self.index = 0;
                                self.previous.style.display =
"none";
                        }

                        if( self.index > 0 ) {
                                self.previous.style.display =
"block";
                        }

                        if( self.index == self.total - 1 ) {
                                self.index = self.total - 1;
                                self.next.style.display =
"none";
                        } else {
                                self.next.style.display =
"block";
                        }

                        self._slideTo( self.index );

                        self._highlightCurrentLink( this );



                        }, false);
                }
}
```

Esempi

- jQuery: slideshow with pagination and controls
- JavaScript: slideshow with pagination and controls

Gestire le dimensioni

Torniamo alle seguente regola CSS per il contenitore interno:

```
.slider-wrapper {
        width: 9999px;
        height: 683px;
        position: relative;
        transition: left 500ms linear;
}
```

9999 pixels non sarà sufficiente se ci sono molte slide e soprattutto se lo slideshow è a tutta pagina. Quindi dobbiamo impostare le dimensioni in modo dinamico basandoci sulla larghezza di ciascuna slide e sul numero complessivo di slide.

Con jQuery è semplice:

```
// Slideshow a tutta pagina

return this.each(function() {
        var $element = $( this ),
                total = $( options.slides ).length;
                //...
                $( options.slides, $element ).width( $(
window ).width() );
                $( options.wrapper, $element ).width( $(
window ).width() * total );
                //...
});
```

Abbiamo usato la larghezza della finestra del browser come base e l'abbiamo moltiplicata per il numero totale di slide.

```
// Slideshow a larghezza fissa
```

```
return this.each(function() {
        var $element = $( this ),
                total = $( options.slides ).length;
                //...

                $( options.wrapper, $element ).width( $(
options.slides ).eq( 0 ).width() * total );
                //...
});
```

In questo caso invece la larghezza di base è data dalla larghezza di una singola slide. In JavaScript avremo:

```
// Slideshow a tutta pagina

Slideshow.prototype = {
        init: function() {
            this.wrapper = this.el.querySelector( ".slider-
wrapper" );
                this.slides = this.el.querySelectorAll(
".slide" );
                //...
                this.total = this.slides.length;

                this.setDimensions();
                this.actions();
        },
        setDimensions: function() {
                var self = this;
                // Larghezza della finestra del browser
                var winWidth = window.innerWidth ||
document.documentElement.clientWidth ||
document.body.clientWidth;
                var wrapperWidth = winWidth * self.total;
                for( var i = 0; i < self.total; ++i ) {
                        var slide = self.slides[i];
                        slide.style.width = winWidth + "px";
                }
```

```
                self.wrapper.style.width = wrapperWidth +
"px";
        },
        //...


};
```

```
// Slideshow a larghezza fissa

Slideshow.prototype = {
        init: function() {
            this.wrapper = this.el.querySelector( ".slider-
wrapper" );
                this.slides = this.el.querySelectorAll(
".slide" );
                //...
                this.total = this.slides.length;

                this.setDimensions();
                this.actions();
        },
        setDimensions: function() {
                var self = this;
                var slideWidth = self.slides[0].offsetWidth;
// Larghezza di una singola slide
                var wrapperWidth = slideWidth * self.total;
                self.wrapper.style.width = wrapperWidth +
"px";
        },
        //...


};
```

Effetti di assolvenza e dissolvenza

Questi effetti sono molto comuni negli slideshow: quando la slide corrente sparisce in dissolvenza, la slide successiva entra in assolvenza e così via. jQuery fornisce i metodi `.fadeIn()` (assolvenza) e `.fadeOut()` (dissolvenza). Purtroppo questi due metodi modificano anche il valore della proprietà CSS `display`, quindi non sono molto indicati se vogliamo creare uno slideshow accessibile.

Possiamo invece usare la proprietà CSS `opacity`. Ecco le impostazioni CSS di base:

```
.slider {
        width: 100%;
        overflow: hidden;
        position: relative;
        height: 400px;
}

.slider-wrapper {
        width: 100%;
        height: 100%;
        position: relative;
}

.slide {
        position: absolute;
        width: 100%;
        height: 100%;
        opacity: 0;
}

.slider-wrapper > .slide:first-child {
        opacity: 1;
}
```

In JavaScript dobbiamo creare una transizione su ciascuna slide:

```
.slide {
        position: absolute;
        width: 100%;
        height: 100%;
        opacity: 0;
        transition: opacity 500ms linear;
}
```

In jQuery se volete comunque usare i metodi `.fadeIn()` e `.fadeOut()`, dovete modificare il codice come segue:

```
.slide {
        position: absolute;
        width: 100%;
        height: 100%;
        display: none;
}

.slider-wrapper > .slide:first-child {
        display: block;
}
```

Ecco il codice jQuery:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {

            options = $.extend({
                        wrapper: ".slider-wrapper",
                        previous: ".slider-previous",
                        next: ".slider-next",
                        slides: ".slide",
                        nav: ".slider-nav",
                        speed: 500,
```

```
                        easing: "linear"

                }, options);

                var slideTo = function( slide, element ) {
                        var $currentSlide = $(
options.slides, element ).eq( slide );

                        $currentSlide.
                        animate({
                                opacity: 1
                        }, options.speed, options.easing ).
                        siblings( options.slides ).
                        css( "opacity", 0 );

                };

                //...
        };

})( jQuery );
```

Quando animiamo la proprietà `opacity` sulla slide corrente dobbiamo anche resettare il suo valore sulle altre slide per evitare sovrapposizioni. In JavaScript avremo:

```
Slideshow.prototype = {
        //...
        _slideTo: function( slide ) {
                var currentSlide = this.slides[slide];
                currentSlide.style.opacity = 1;

                for( var i = 0; i < this.slides.length; i++ )
{
                        var slide = this.slides[i];
                        if( slide !== currentSlide ) {
                                slide.style.opacity = 0;
                        }
```

```
                }
        },
        //...
};
```

Esempi

- jQuery: full width slideshow with fading effect
- JavaScript: slideshow with fading effect

Elementi multimediali: video

Possiamo anche includere elementi multimediali nei nostri slideshow. Ecco un esempio con i video di Vimeo:

```
<div class="slider-wrapper"><!-- contenitore interno -->
                <div class="slide">
                        <iframe
src="https://player.vimeo.com/video/109608341?
title=0&amp;byline=0&amp;portrait=0" width="1024"
height="626" frameborder="0" webkitallowfullscreen
mozallowfullscreen allowfullscreen></iframe>
                </div><!-- slides -->
                <div class="slide">
                        <iframe
src="https://player.vimeo.com/video/102570343?
title=0&amp;byline=0&amp;portrait=0" width="1024"
height="576" frameborder="0" webkitallowfullscreen
mozallowfullscreen allowfullscreen></iframe>
                </div>
                <div class="slide">
                        <iframe
src="https://player.vimeo.com/video/97620325?
title=0&amp;byline=0&amp;portrait=0" width="1024"
height="576" frameborder="0" webkitallowfullscreen
mozallowfullscreen allowfullscreen></iframe>
```

```
                </div>
</div>
```

Se vogliamo creare uno slideshow di video a tutta pagina, dobbiamo modificare gli stili CSS come segue:

```
html, body {
        margin: 0;
        padding: 0;
        height: 100%;
        min-height: 100%; /* Altezza al 100% */
}
.slider {
        width: 100%;
        overflow: hidden;
        height: 100%;
        min-height: 100%; /* Altezza e larghezza al 100% */
        position: absolute; /* Posizionamento assoluto */
}

.slider-wrapper {
        width: 100%;
        height: 100%; /* Altezza e larghezza al 100% */
        position: relative;
}

.slide {
        position: absolute;
        width: 100%;
        height: 100%;
}

.slide iframe {
        display: block; /* Elemento di blocco */
        position: absolute; /* Posizionamento assoluto */
        width: 100%;
        height: 100%; /* Altezza e larghezza al 100% */
}
```

Esempi

- jQuery: full page video slideshow
- JavaScript: full page video slideshow

Slideshow automatici

Gli slideshow automatici funzionano per mezzo dei timer JavaScript (`setTimeout()` o `setInterval()`). Il timer andrà ad incrementare il nostro contatore che useremo per selezionare le slide. Quando il valore del contatore sarà uguale al numero totale di slide dovremo reimpostarlo su 0 per far ripartire la sequenza.

Inoltre la durata dell'intervallo del timer dovrà essere uguale alla durata complessiva dell'animazione per evitare inutili code. Un altro aspetto è quello di permettere all'utente di mettere in pausa la sequenza quando il mouse è sopra lo slideshow e di conseguenza farla ripartire quando il mouse si allontana dallo slideshow.

In jQuery possiamo scrivere:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
                options = $.extend({
                        slides: ".slide",
                        speed: 3000,
                        easing: "linear"

                }, options);

                var timer = null; // Il timer
                var index = 0;    // Il contatore

                var slideTo = function( slide, element ) {
                        var $currentSlide = $(
```

```
options.slides, element ).eq( slide );

                        $currentSlide.stop( true, true ).
                        animate({
                                opacity: 1
                        }, options.speed, options.easing ).
                        siblings( options.slides ).
                        css( "opacity", 0 );

                };

                var autoSlide = function( element ) {
                        // Inizia la sequenza
                        timer = setInterval(function() {
                                index++; // Incrementa il
contatore
                                if( index == $(
options.slides, element ).length ) {
                                        index = 0; // Resetta
il contatore
                                }
                                slideTo( index, element );
                        }, options.speed); // Stesso
intervallo del metodo .animate()
                };

                var startStop = function( element ) {
                        element.hover(function() { // Ferma
la sequenza su hover
                                clearInterval( timer );
                                timer = null;
                        }, function() {
                                autoSlide( element ); // Fa
ripartire la sequenza
                        });
                };

                return this.each(function() {
                        var $element = $( this );

                                autoSlide( $element );
                                startStop( $element );
```

```
                });
        };

})( jQuery );
```

Usiamo il metodo .stop() perché non vogliamo nessuna coda di animazioni.

In JavaScript dobbiamo creare una transizione CSS con la durata appropriata:

```
.slide {
        transition: opacity 3s linear; /* 3 secondi = 3000
millesimi di secondo */
}
```

Quindi possiamo scrivere:

```
(function() {

        function Slideshow( element ) {
                this.el = document.querySelector( element );
                this.init();
        }

        Slideshow.prototype = {
                init: function() {
                        this.slides =
this.el.querySelectorAll( ".slide" );
                        this.index = 0; // Il contatore
                        this.timer = null;  // Il timer

                        this.action();
                        this.stopStart();
                },
                _slideTo: function( slide ) {
                        var currentSlide =
```

```
this.slides[slide];
                        currentSlide.style.opacity = 1;

                        for( var i = 0; i <
this.slides.length; i++ ) {
                                var slide = this.slides[i];
                                if( slide !== currentSlide )
{
                                        slide.style.opacity =
0;
                                }
                        }
                },
                action: function() {
                        var self = this;
                        // Inizia la sequenza
                        self.timer = setInterval(function() {
                                self.index++; // Incrementa
il contatore
                                if( self.index ==
self.slides.length ) {
                                        self.index = 0; //
Resetta il contatore
                                }
                                self._slideTo( self.index );

                        }, 3000); // Stesso intervallo della
transizione CSS
                },
                stopStart: function() {
                        var self = this;
                        // Ferma la sequenza su hover
                        self.el.addEventListener(
"mouseover", function() {
                                clearInterval( self.timer );
                                self.timer = null;

                        }, false);
                        // Fa ripartire la sequenza
                        self.el.addEventListener( "mouseout",
function() {
                                self.action();
```

```
                        }, false);
                }


        };
})();
```

Esempi

- jQuery: automatic slideshow
- JavaScript: automatic slideshow

Navigazione da tastiera

Alcuni slideshow forniscono all'utente la possibilità di navigare le slide tramite la tastiera. In pratica si tratta di gestire l'evento `keydown` in JavaScript. Questo tipo di evento possiede la proprietà `keyCode` per l'oggetto `Event`. Questa proprietà restituisce un codice numerico che ci permette di stabilire quale tasto è stato premuto (consultate questa pagina per un elenco completo dei codici).

Se vogliamo associare ai tasti freccia destra e sinistra le azioni dei pulsanti "Precedente" e "Successivo", in jQuery possiamo scrivere:

```
$( "body" ).on( "keydown", function( e ) {
        var code = e.keyCode;
        if( code == 39 ) { // Freccia destra
                $next.trigger( "click" );
        }
        if( code == 37 ) { // Freccia sinistra
                $previous.trigger( "click" );
        }

});
```

Abbiamo lanciato l'evento `click` sul pulsante corrispondente. In JavaScript la procedura è la stessa ma non abbiamo il metodo `.trigger()` di jQuery: dobbiamo usare il metodo dispatchEvent() nel contesto degli eventi del mouse:

```
document.body.addEventListener( "keydown", function( e ) {
        var code = e.keyCode;
        var evt = new MouseEvent( "click" );  // Evento click

        if( code == 39 ) { // Freccia destra
                self.next.dispatchEvent( evt );
        }
        if( code == 37 ) { // Freccia sinistra
                self.previous.dispatchEvent( evt );
        }

}, false);
```

Esempi

- jQuery: slideshow with keyboard navigation
- JavaScript: slideshow with keyboard navigation

Callback

I callback sono funzioni che vengono invocate quando si verifica un determinato evento o quando un'azione ha luogo. Nel contesto degli slideshow possiamo ad esempio eseguire un'azione dopo l'animazione principale di ogni slide.

In jQuery possiamo implementare i callback in questo modo:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
```

```
                options = $.extend({
                        //...
                        callback: function() {}

                }, options);

                var slideTo = function( slide, element ) {
                        var $currentSlide = $(
options.slides, element ).eq( slide );

                        $currentSlide.
                        animate({
                                opacity: 1
                        }, options.speed,
                           options.easing,
                           // Callback sulla slide corrente
                           options.callback( $currentSlide )
).
                           siblings( options.slides ).
                           css( "opacity", 0 );

                };

        //...
    };
})( jQuery );
```

In questo caso il callback è associato con la funzione di callback del metodo `.animate()` e accetta la slide corrente come suo argomento. Ecco come può essere usato:

```
$(function() {
        $( "#main-slider" ).slideshow({
                callback: function( slide ) {
                        var $wrapper = slide.parent();
                        // Mostra la didascalia corrente
                        $wrapper.find( ".slide-caption"
).hide();
                        slide.find( ".slide-caption" ).show(
```

```
"slow" );
                }

        });
});
```

In JavaScript avremo:

```
(function() {

        function Slideshow( element, callback ) {
                this.callback = callback || function() {}; //
Il nostro callback
                this.el = document.querySelector( element );
                this.init();
        }

        Slideshow.prototype = {
                init: function() {
                        //...
                        this.slides =
this.el.querySelectorAll( ".slide" );
                        //...

                        //...
                },
                _slideTo: function( slide ) {
                        var self = this;
                        var currentSlide =
self.slides[slide];
                        currentSlide.style.opacity = 1;

                        for( var i = 0; i <
self.slides.length; i++ ) {
                                var slide = self.slides[i];
                                if( slide !== currentSlide )
{
                                        slide.style.opacity =
0;
```

```
                                }
                        }
                        setTimeout( self.callback(
currentSlide ), 500 );
                        // Il callback viene invocato quando
la transizione è completa
                }

        };
    //
})();
```

Il callback viene usato in questo modo:

```
document.addEventListener( "DOMContentLoaded", function() {

                var slider = new Slideshow( "#main-slider",
function( slide ) {
                        var wrapper = slide.parentNode;

                        // Mostra la didascalia corrente

                        var allSlides =
wrapper.querySelectorAll( ".slide" );
                        var caption = slide.querySelector(
".slide-caption" );
                        caption.classList.add( "visible" );

                   for( var i = 0; i < allSlides.length; ++i
) {
                          var sld = allSlides[i];
                          var cpt = sld.querySelector(
".slide-caption" );
                   if( sld !== slide ) {
                        cpt.classList.remove( "visible" );
                   }
                }
```

```
        });

});
```

Esempi

- jQuery: slideshow with callback
- JavaScript: slideshow with callback

Usare le API

Possiamo anche utilizzare le API JSON o JSONP di YouTube, Vimeo, Flickr ed altri web service per creare le slide del nostro slideshow. Dato che la risposta del server remoto può non essere immediata, dobbiamo creare un indicatore di caricamento per indicare che il download delle risorse è in corso.

```
<div class="slider" id="main-slider"><!-- contenitore esterno
-->
        <div class="slider-wrapper"><!-- contenitore interno
-->

        </div>
        <div class="slider-nav"><!-- "Precedente" e
"Successivo" -->
                <button class="slider-
previous">Precedente</button>
                <button class="slider-
next">Successivo</button>
        </div>
        <div id="spinner"></div><!-- indicatore di
caricamento -->
</div>
```

I passi da seguire sono:

1. Reperire i dati remoti
2. Nascondere l'indicatore di caricamento
3. Effettuare il parsing dei dati
4. Costruire il contenuto HTML
5. Creare le slide
6. Gestire lo slideshow.

Possiamo ad esempio reperire i video di un utente da YouTube. In jQuery il codice è il seguente:

```
(function( $ ) {
        $.fn.slideshow = function( options ) {
                options = $.extend({
                        wrapper: ".slider-wrapper",
                    //...
                        loader: "#spinner",
                        //...
                        limit: 5,
                        username: "learncodeacademy"

                }, options);

                //...

                var getVideos = function() {
                        // Prende i video da YouTube
                        var ytURL =
"https://gdata.youtube.com/feeds/api/videos?alt=json&author="
+ options.username + "&max-results=" + options.limit;
                        $.getJSON( ytURL, function( videos )
{ // Oggetto JSON
                                $( options.loader ).hide();
// Nasconde l'indicatore di caricamento
                                var entries =
videos.feed.entry;
                                var html = "";
                                for( var i = 0; i <
entries.length; ++i ) { // Parsing e creazione della stringa
HTML
```

```
                                        var entry =
entries[i];
                                        var idURL =
entry.id.$t;
                                        var idVideo =
idURL.replace( "http://gdata.youtube.com/feeds/api/videos/",
"" );
                                        var ytEmbedURL =
"https://www.youtube.com/embed/" + idVideo + "?
rel=0&showinfo=0&controls=0";

                                        html += "<div
class='slide'>";
                                        html += "<iframe
src='" + ytEmbedURL + "' frameborder='0' allowfullscreen>
</iframe>";
                                        html += "</div>";
                                }

                                $( options.wrapper ).html(
html ); // Creazione delle slide

                        });


                };

                return this.each(function() {
                        //...
                        getVideos();

                        // Gestione dello slideshow

                });
    };
})( jQuery );
```

In JavaScript dobbiamo creare un metodo per reperire il feed JSON:

```
(function() {

	function Slideshow( element ) {
		this.el = document.querySelector( element );
		this.init();
	}

	Slideshow.prototype = {
		init: function() {
			this.wrapper = this.el.querySelector(
".slider-wrapper" );
			this.loader = this.el.querySelector(
"#spinner" );
			//...
			this.limit = 5;
			this.username = "learncodeacademy";


		},
		_getJSON: function( url, callback ) {
			callback = callback || function() {};

			var request = new XMLHttpRequest();

			request.open( "GET", url, true );
			request.send( null );

			request.onreadystatechange =
function() {
				if ( request.status == 200 &&
request.readyState == 4 ) {

					var data =
JSON.parse( request.responseText ); // Oggetto JSON



					callback( data );

				} else {
					console.log(
```

```
request.status );

                                }
                        };

                },
                //...

        };
})();
```

Quindi le routine sono simili:

```
(function() {

        function Slideshow( element ) {
                this.el = document.querySelector( element );
                this.init();
        }

        Slideshow.prototype = {
                init: function() {
                        this.wrapper = this.el.querySelector(
".slider-wrapper" );
                        this.loader = this.el.querySelector(
"#spinner" );
                        //...
                        this.limit = 5;
                        this.username = "learncodeacademy";

                        this.actions();

                },
                _getJSON: function( url, callback ) {
                        callback = callback || function() {};

                        var request = new XMLHttpRequest();

                        request.open( "GET", url, true );
```

```
                        request.send( null );

                        request.onreadystatechange =
function() {
                                if ( request.status == 200 &&
request.readyState == 4 ) {

                                        var data =
JSON.parse( request.responseText ); // Oggetto JSON



                                        callback( data );

                                } else {
                                        console.log(
request.status );

                                }
                        };

                },
                //...
                getVideos: function() {
                        var self = this;
                        // Video di YouTube
                        var ytURL =
"https://gdata.youtube.com/feeds/api/videos?alt=json&author="
+ self.username + "&max-results=" + self.limit;

                        self._getJSON( ytURL, function(
videos ) { // Oggetto JSON
                                var entries =
videos.feed.entry;
                                var html = "";
                                self.loader.style.display =
"none";  // Nasconde l'indicatore di caricamento

                                for( var i = 0; i <
entries.length; ++i ) { // Parsing dei dati
                                        var entry =
entries[i];
                                        var idURL =
```

```
entry.id.$t;
                                        var idVideo =
idURL.replace( "http://gdata.youtube.com/feeds/api/videos/",
"" );
                                        var ytEmbedURL =
"https://www.youtube.com/embed/" + idVideo + "?
rel=0&amp;showinfo=0&controls=0";

                                        html += "<div
class='slide'>";
                                        html += "<iframe
src='" + ytEmbedURL + "' frameborder='0' allowfullscreen>
</iframe>";
                                        html += "</div>";
                                }

                                self.wrapper.innerHTML =
html; // Creazione delle slide

                        });

                },
                actions: function() {
                        var self = this;

                        self.getVideos();

                        // Gestione dello slideshow
                }



        };
})();
```

Esempi

- jQuery: using external APIs
- JavaScript: using external APIs

Conclusione

Gli slideshow rappresentano un'eccellente opportunità di migliorare l'interfaccia utente, ma senza la conoscenza delle basi del loro sviluppo non è possibile progettare alcun tipo di implementazione.

Esempi completi

GitHub