# Validare HTML5 in WordPress

La validazione di HTML5 in WordPress non è così semplice come può sembrare in apparenza. Anche se HTML5 ha di fatto rinunciato ad un approccio basato sulla DTD, la sua grammatica formale viene di fatto definita dalle specifiche (ancora in stato di bozza). C'è da dire che la validazione HTML5 è ancora in fase sperimentale, quindi i risultati potrebbero cambiare nel tempo. Detto questo, ecco alcuni consigli pratici.

## Codifica delle entità e dei caratteri

Dovreste sempre codificare tutte le entità ed i caratteri usati nel vostro tema o nei vostri post. Per farlo, dovreste sempre verificare che i caratteri inseriti siano di fatto contemplati dalla codifica usata nel vostro sito (di solito UTF-8).

Uno degli errori più frequenti riguarda i plugin dei social network che utilizzano URL con parametri. Ad esempio:

```
$html = '<iframe allowtransparency="true"
frameborder="0" scrolling="no"
src="http://platform.twitter.com/widgets/tweet_button
.html?';
                $html .= 'url=' . $link . '&text=' .
$title . '&count=' . $layout . '&via=' . $via . '"
style="width: 130px; height: 20px;"></iframe>';
```

In questo caso abbiamo il separatore & che viene inserito letteralmente. Andrebbe quindi codificato come `&`. Come regola di massima, dovreste anche usare la funzione PHP urlencode() per questo tipo di URL.

# Elementi e attributi non validi

SGML (l'antenato di HTML5) prevede che un elemento possa presentare solo una determinata serie di attributi. Questa caratteristica è ancora più marcata in XML e XHTML, dove di fatto la DTD prevede una `ATTLIST` di attributi permessi per ciascun elemento.

In HTML5 vale la stessa cosa. Ad esempio, nel codice precedente l'attributo `allowtransparency` non è di fatto valido. Bisognerebbe semplicemente usare:

```
<iframe frameborder="0" scrolling="no" src="">
</iframe>
```

Un caso tipico è la marcatura generata dai widget o dai plugin. Molto spesso i widget o i plugin generano marcatura non valida, come ad esempio:

```
function output_header_placeholder() {
                echo '<meta
id="syntaxhighlighteranchor" name="syntaxhighlighter-
version" content="' . esc_attr( $this->pluginver ) .
'" />' . "\n";
}
```

Per quanto riguarda invece gli elementi, essi generalmente non sono considerati validi quando appaiono in un contesto sbagliato o sono male annidati. Per esempio, l'output predefinito dei widget di WordPress genera la seguente struttura:

```
<ul>
        <h3>...</h3>
</ul>
```

Un elemento `h3` non può essere direttamente contenuto in un elemento `ul`. Dovreste per questo motivo modificare sempre la struttura predefinita di questi componenti o definire dei widget personalizzati.

Infine, alcuni elementi (come `acronym`) sono deprecati in HTML5, quindi se ne sconsiglia l'uso.

## Sezioni PCDATA e CDATA

In XML, una sezione di tipo `PCDATA` contiene sequenze di caratteri interpretati come marcatura, mentre una sezione `CDATA` contiene del normale testo.

Consideriamo questo codice:

```
<script>
(function($) {

        $(function() {

                if($(window).width() > 1024) {

                        //...


                }

        });
```

```
})(jQuery);
</script>
```

L'errore sta nel carattere > che viene interpretato come l'inizio di un tag, ossia come PCDATA. In altre parole, se il vostro codice JavaScript contiene caratteri che possono essere interpretati come marcatura, dovreste sempre racchiuderlo tra commenti HTML:

```
<script>
<!--
(function($) {

        $(function() {

                if($(window).width() > 1024) {

                        //...


                }

        });

})(jQuery);
-->
</script>
```

Una soluzione migliore, che consiglio, è usare sempre file JavaScript esterni (laddove è possibile).