Lezione 2.1:

Elementi di Javascript, Wordpress e personalizzazioni

Indice dei Contenuti

Introduzione

JavaScript è un linguaggio di programmazione che rende le pagine web interattive. Mentre l'HTML definisce il contenuto di una pagina e il CSS ne definisce l'aspetto, JavaScript aggiunge comportamenti dinamici consentendo di interagire con l'utente. JavaScript è l'ambiente di interazione preferito per generare scambio di dati e le interfacce web.

Obiettivi di apprendimento

Al termine di questa lezione, sarai in grado di:

- Conoscere le interazioni di base da generare tramite JavaScript
- Identificare le tipologie di dati e catalogare l'informazione su cui generare interazioni

- Generare pulsanti e azioni di base per gestire i dati tramite l'interazione tra HTML e JavaScript
- Avviare e installare una soluzione Wordpress
- Ottimizzare la soluzione installata per renderla online in modo sicuro e stabile

1. Introduzione a JavaScript

JavaScript è il principale linguaggio di programmazione per lo sviluppo di web applications. Sempre più diffuso, tocca ormai gli ambiti mobile, server e desktop.

Concetti di base

1. **Variabili**: sono i contenitori per memorizzare i dati

```
let nome = "Mario";

const età = 30;

var città = "Roma"; // metodo più vecchio, meglio usare let o const
```

2. **Tipi di dati**: la panoramica dei diti di dati gestiti in JavaScript include

String (Stringa)

- Testo racchiuso tra apici singoli, doppi o backtick:

```
let nome = "Mario";

let cognome = 'Rossi';

let frase = `Ciao, mi chiamo ${nome}`; // Template string con backtick
```

Number (Numero)

- Numeri interi o decimali:

```
let intero = 42;

let decimale = 3.14;
```


by: Ulises Gabriel Miranda – mail@ulises.it

```
let negativo = -10;

let esponenziale = 1.23e6; // 1230000
```

Boolean (Booleano)

- Valore logico vero/falso

```
let vero = true;

let falso = false;
```

Undefined (Indefinito)

- Rappresenta una variabile dichiarata ma senza valore assegnato

```
let variabile;  // valore è undefined
```

Undefined (Indefinito)

- Rappresenta una variabile dichiarata ma senza valore assegnato

```
let variabile;  // valore è undefined
```

Null

- Rappresenta l'assenza intenzionale di un valore

```
let vuoto = null;
```

BigInt

- Per dichiarare o specificare (dopo calcoli) numeri interi molto grandi

```
let numeroGrande = 9007199254740991n;

let altroBigInt = BigInt("9007199254740991");
```


by: Ulises Gabriel Miranda – mail@ulises.it

Symbol

- Valore unico e immutabile, spesso usato come chiave di proprietà

```
let id = Symbol("id");
```

Tipi di dati strutturati (non primitivi)

- **Object (Oggetto)**
  - Collezione di coppie chiave-valore

```
let persona = {
 nome: "Mario",
 età: 30,
 indirizzo: {
  via: "Via Roma",
  città: "Milano"
 }
};
```

- **Array**
  - Lista ordinata di valori

```
let frutta = ["mela", "banana", "kiwi"];
let misto = [1, "due", true, {chiave: "valore"}];
```

- **Function (Funzione)**
  - In JavaScript, le funzioni sono trattate come oggetti

```
function somma(a, b) {
 return a + b;
}
// Funzione come espressione
```


by: Ulises Gabriel Miranda – mail@ulises.it

```
const moltiplica = function(a, b) {
 return a * b;
};

// Arrow function
const dividi = (a, b) => a / b;
```

- **Date (Data)**

```
let oggi = new Date();
```

- **RegExp (Espressione regolare)**

```
let pattern = /\d+/;          // Corrisponde a una o più cifre
let altroPattern = new RegExp("\\d+");
```

- **Map**

Collezione di coppie chiave-valore dove le chiavi possono essere di qualsiasi tipo:

```
let mappa = new Map();
mappa.set("nome", "Luigi");
mappa.set(1, "numero uno");
```

- **Set**

Collezione di valori unici:

```
let insieme = new Set();
insieme.add("valore1");
insieme.add("valore2");
```


by: Ulises Gabriel Miranda – mail@ulises.it

- **WeakMap e WeakSet**

Weakset:

Versioni "deboli" di Map e Set che permettono la garbage collection

Il motore JavaScript mantiene un valore in memoria fino a che questo risulta accessibile (e potrebbe potenzialmente essere utilizzato).

Ad esempio:

```
let john = { name: "John" };

// l'oggetto è accessibile, john è un suo riferimento


// sovrascriviamo il riferimento

john = null;


// l'oggetto verrà rimosso dalla memoria
```

Solitamente, le proprietà di un oggetto o gli elementi di un array o di qualsiasi altra struttura dati vengono considerati accessibili fino a che questi rimangono mantenuti in memoria.

Ad esempio, se inseriamo un oggetto in un array, fino a che l'array rimane "vivo", anche l'oggetto rimarrà in memoria, anche se non sono presenti riferimenti.

Come nell'esempio:

```
let john = { name: "John" };

let array = [ john ];

john = null;     // sovrascriviamo il riferimento
```


by: Ulises Gabriel Miranda – mail@ulises.it

```
// john è memorizzato all'interno dell'array
// quindi non verrà toccato dal garbage collector
// possiamo estrarlo tramite array[0]
```

O, se utilizziamo un oggetto come chiave in una Map, fino a che la Map esiste, anche l'oggetto esisterà. Occuperà memoria e non potrà essere ripulito dal garbage collector.

Ad esempio:

```
let john = { name: "John" };

let map = new Map();
map.set(john, "...");

john = null;     // sovrascriviamo il riferimento

// john viene memorizzato all'interno di map,
// possiamo estrarlo utilizzando map.keys()
```

Weakmap

La prima differenza tra Map e WeakMap è che le chiavi devono essere oggetti, non valori primitivi:

```
let weakMap = new WeakMap();
let obj = {};
weakMap.set(obj, "ok");                 // funziona

// non possiamo utilizzare una stringa come chiave
weakMap.set("test", "Whoops");     // Errore, perché "test" non è un oggetto
```


by: Ulises Gabriel Miranda – mail@ulises.it

Ora, se utilizziamo un oggetto come chiave, e non ci sono altri riferimenti a quell'oggetto – questo verrà rimosso dalla memoria (e dalla map) automaticamente.

```
let john = { name: "John" };

let weakMap = new WeakMap();
weakMap.set(john, "...");

john = null;                    // sovrascriviamo il riferimento

// john è stato rimosso dalla memoria!
```

Confrontiamolo con l'esempio di Map visto sopra. Ora, se john esiste solo come chiave della WeakMap – verrà eliminato automaticamente dalla map (e anche dalla memoria).

WeakMap non supporta gli iteratori e i metodi keys(), values(), entries(), quindi non c'è alcun modo di ottenere tutte le chiavi o i valori tramite questi metodi.

WeakMap possiede solamente i seguenti metodi:

```
weakMap.get(key)
weakMap.set(key, value)
weakMap.delete(key)
weakMap.has(key)
```

Perché questa limitazione? Per ragioni tecniche. Se un oggetto ha perso tutti i riferimenti (come john nel codice sopra), allora verrà automaticamente eliminato. Ma tecnicamente non è specificato esattamente quando avverrà la pulizia.

Sarà il motore JavaScript a deciderlo. Potrebbe decidere di effettuare subito la pulizia della memoria oppure aspettare più oggetti per eliminarli in blocco. Quindi, tecnicamente il numero degli elementi di una WeakMap non è conosciuto. Il motore potrebbe già aver effettuato la pulizia oppure no, o averlo fatto solo parzialmente. Per questo motivo, i metodi che accedono a WeakMap per intero non sono supportati.

Dove potremmo avere bisogno di una struttura simile?

Il principale campo di applicazione di WeakMap è quello di un *additional data storage*.

Se stiamo lavorando con un oggetto che "appartiene" ad un altro codice, magari una libreria di terze parti, e vogliamo memorizzare alcuni dati associati ad esso, che però dovrebbero esistere solamente finché l'oggetto esiste – allora una WeakMap è proprio ciò di cui abbiamo bisogno.

Controllare il tipo di dati

Si può verificare il tipo di un valore usando l'operatore typeof:

```
console.log(typeof "ciao");              // "string"
console.log(typeof 42);                  // "number"
console.log(typeof true);                // "boolean"
console.log(typeof undefined);           // "undefined"
console.log(typeof null);                // "object" (questo è un errore storico di JavaScript)
console.log(typeof {});                  // "object"
console.log(typeof []);                  // "object" (gli array sono oggetti in JavaScript)
console.log(typeof function(){});        // "function"
```

Per controllare se un valore è un array, va utilizzato:

Array.isArray(), ovvero:

by: Ulises Gabriel Miranda – mail@ulises.it

```
console.log(Array.isArray([]));          // true
console.log(Array.isArray({}));          // false
```

NB. Guida specifica **JavaScript, Gli strumenti di lavoro (**da HTML.it a questo **LINK)**

Ambiente Wordpress

Wordpress è uno dei sistemi di gestione di contenuti (CMS = Content Management System), più noti per la sua versatilità e la possibilità di personalizzare le configurazioni in molteplici contesti di utilizzo. Questo ambiente comprende vari elementi, che insieme permettono di creare e mantenere un sito web dinamico e interattivo.

Dinamico → si adatta alle necessità mutevoli del contesto di utilizzo

Interattivo → consente interazioni basate sullo sviluppo e perfezionamento condiviso dei contenuti

Aneddoti: L'origine

WordPress è nato nel 2003 da un'idea di Matt Mullenweg e Mike Little. È stato sviluppato come fork di b2/cafelog, un precedente progetto di blogging.

Com'è nato WordPress?

- Nel 2002, Matt Mullenweg, uno studente universitario, installò b2/cafelog per uso personale.
- Quando il creatore originale di b2/cafelog non ha più aggiornato il progetto, Matt ha creato un nuovo branch di b2 su SourceForge.
- Il 1° aprile 2003, Christine Tremoulet, un'amica di Matt, gli ha suggerito di chiamarlo WordPress.
- Il 27 maggio 2003, Matt Mullenweg e Mike Little hanno pubblicato la prima versione di WordPress.

Come si è evoluto WordPress?

- WordPress è nato come piattaforma per blog, ma grazie alla sua versatilità e facilità d'uso è diventato uno strumento molto popolare per creare siti web di ogni tipo.
- WordPress è un software open source, quindi è libero di essere utilizzato e modificato.

- WordPress è diventato la soluzione più popolare per creare qualsiasi tipo di sito web.

WORDPRESS.org (Open Source) → **WORDPRESS.com** (Servizio su Licenza)

Componenti del sistema

Esploreremo i componenti chiave di un ambiente Wordpress e come interagiscono tra loro per fornire una piattaforma robusta per la creazione di contenuti online. Questi elementi devono essere pianificati per consentire un flusso di lavoro e di interazioni in armonia a seconda dello scopo del sito web attivato.

1. Server Web

Il server web è il luogo in cui risiede il tuo sito WordPress. Può essere un server fisico o un server virtuale, e può essere ospitato localmente o su un servizio di hosting. I server più comuni per WordPress sono Apache e Nginx. L'ambiente può essere generato anche in locale e sfruttabile online tramite impostazioni IP e disponibilità dedicata di risorse.

2. Database

WordPress utilizza un database per memorizzare tutte le informazioni relative al sito, come post, pagine, commenti e impostazioni. MySQL è il sistema di gestione di database più utilizzato con WordPress, anche se è possibile utilizzare MariaDB come alternativa.

3. File di WordPress

I file di WordPress comprendono il core del CMS, i temi e i plugin. Il core è il codice sorgente di WordPress, mentre i temi determinano l'aspetto visivo del sito e i plugin aggiungono funzionalità extra.

4. Strumenti di Sviluppo

Per gestire e sviluppare un sito WordPress, è utile avere a disposizione strumenti come editor di codice, software di gestione del database e strumenti di debug. Questi strumenti aiutano a personalizzare e ottimizzare il sito.

Installare e attivare un ambiente Wordpress

WordPress è uno dei sistemi di gestione dei contenuti (CMS) più popolari al mondo, che permette di creare siti web di vario tipo senza necessariamente conoscere linguaggi di programmazione complessi. Ecco una guida passo-passo per iniziare con WordPress in locale e poi scegliere i plugin giusti per il tuo progetto.

1. Installazione in ambiente locale

Per sviluppare un sito WordPress in locale (sul tuo computer), hai bisogno di:

1. **Un server web locale**
2. **Un database MySQL**
3. **PHP**

La soluzione più semplice è installare un pacchetto "all-in-one" che include tutti questi componenti.

Opzioni per l'ambiente di sviluppo locale:

by: Ulises Gabriel Miranda – mail@ulises.it

- **XAMPP** (Windows, Mac, Linux)

  XAMPP (X sta per Cross-platform, A per Apache server, M da MySQL, mentre la doppia P per PHP e Per) è un insieme di software gratuito e open source che permette di creare un server web locale sul proprio computer. È utile per testare progetti web prima di caricarli su un server remoto.

  Caratteristiche:

  - È disponibile per i sistemi operativi Windows, Linux, Mac e Solaris
  - È facile da installare e utilizzare
  - È composto da Apache HTTP Server, MySQL o MariaDB, PHP e Perl
  - Include tutti gli strumenti necessari per utilizzare i linguaggi di programmazione PHP e Perl
  - Permette di sviluppare e testare applicazioni web in locale

  Come si usa:

  - Si installa scaricando e avviando il file di installazione
  - Per accedere a WordPress da XAMPP, si può:
    1. Selezionare il pulsante Admin in corrispondenza di Apache nel pannello di Xampp
    2. In alternativa, digitare l'indirizzo localhost sul browser
    3. Digitare la cartella del sito

  Perché si usa

  XAMPP è particolarmente usato per scopi di testing nella programmazione web. Permette di testare i progetti in locale e di trasferirli poi comodamente sui sistemi effettivi.

- **MAMP** (Mac, Windows): **M**ac OS X, **A**pache, **M**ySQL, e **P**HP

Altre installazioni dedicate:

- **Local by Flywheel** (soluzione dedicata a WordPress)
- **DevKinsta** (ambiente specifico per WordPress)

Istruzioni dettagliate usando Local by *Flywheel* (consigliato per principianti):

1. **Scarica e installa Local** da localwp.com
2. **Crea un nuovo sito:**
    - Avvia Local e clicca su "Create a new site"
    - Assegna un nome al sito (es. "Mio Progetto WordPress")
    - Scegli la cartella dove salvare i file del sito
3. **Configura l'ambiente:**
    - Scegli "Preferred" per un'installazione rapida
    - Oppure seleziona "Custom" per configurare versioni specifiche di PHP, MySQL e server web
4. **Configura WordPress:**
    - Inserisci username, password e email per l'account amministratore
    - Clicca su "Add Site" e attendi il completamento dell'installazione
5. **Accedi al tuo sito WordPress:**
    - Una volta completata l'installazione, clicca su "Admin" per accedere al pannello di amministrazione
    - Oppure clicca su "View Site" per vedere il frontend del sito

2. Analisi delle funzionalità necessarie per il tuo sito

Prima di installare plugin, è importante definire chiaramente quali funzionalità necessiti per il tuo progetto. Vediamo alcune categorie comuni:

Tipologie di siti web e funzionalità relative:

Sito aziendale / Vetrina:

- Pagine informative
- Moduli di contatto
- Integrazione social media
- SEO
- Condivisione sui social

Blog:

- Sistema di commenti
- Categorie e tag
- Newsletter
- Contenuti correlati
- Condivisione social

E-commerce:

- Catalogo prodotti
- Carrello e checkout
- Gateway di pagamento
- Gestione inventario
- Spedizioni

Portfolio:

- Gallerie di immagini
- Caroselli
- Filtri per progetti

- Lightbox

Membership / Community:

- Registrazione utenti
- Aree riservate
- Forum/discussioni
- Contenuti a pagamento

3. Selezione e installazione dei plugin essenziali

Plugin fondamentali (quasi sempre necessari e però la lista può variare a seconda dell'evoluzione delle caratteristiche / versioni di Wordpress e dei plugin stessi)

1. **Yoast SEO o Rank Math**
    - Ottimizzazione SEO
    - Installazione: Dashboard → Plugin → Aggiungi nuovo → Cerca "Yoast SEO" → Installa → Attiva
2. **Wordfence Security**
    - Protezione del sito
    - Installazione: Dashboard → Plugin → Aggiungi nuovo → Cerca "Wordfence" → Installa → Attiva
3. **WP Super Cache o W3 Total Cache**
    - Miglioramento performance
    - Installazione: Dashboard → Plugin → Aggiungi nuovo → Cerca "WP Super Cache" → Installa → Attiva
4. **UpdraftPlus**
    - Backup automatizzati

by: Ulises Gabriel Miranda – mail@ulises.it

    - Installazione: Dashboard → Plugin → Aggiungi nuovo → Cerca "UpdraftPlus" → Installa → Attiva

5. **Contact Form 7 o WPForms**
    - Creazione moduli di contatto
    - Installazione: Dashboard → Plugin → Aggiungi nuovo → Cerca "Contact Form 7" → Installa → Attiva

Plugin specifici per tipologia di sito:

Per E-commerce:

- **WooCommerce**
    - Installazione: Dashboard → Plugin → Aggiungi nuovo → Cerca "WooCommerce" → Installa → Attiva
    - Segui la procedura guidata di configurazione

Per Blog:

- **Akismet Anti-Spam**
- **Monarch** o **Social Warfare** (per condivisione social)

Per Portfolio:

- **Elementor** o **Divi Builder** (page builder)
- **NextGEN Gallery** o **Envira Gallery**

Per Membership:

- **MemberPress** o **Paid Memberships Pro**
- **BuddyPress** (per funzionalità social)

4. Metodologia per valutare i plugin

Prima di installare un plugin, considera questi aspetti:

1. **Aggiornamenti recenti:** Verifica quando è stato aggiornato l'ultima volta
2. **Valutazioni e recensioni:** Controlla le stelle e leggi i commenti
3. **Compatibilità:** Assicurati che sia compatibile con la tua versione di WordPress
4. **Supporto:** Verifica se gli sviluppatori rispondono alle domande
5. **Documentazione:** Controlla se è ben documentato

5. Configurazione iniziale del sito

Dopo aver installato WordPress e i plugin essenziali:

1. **Imposta le impostazioni generali:**
    - Dashboard → Impostazioni → Generali
    - Configura titolo sito, slogan, email, ecc.
2. **Scegli un tema:**
    - Dashboard → Aspetto → Temi → Aggiungi nuovo
    - Scegli un tema che si adatti alle tue esigenze
3. **Configura i plugin installati:**
    - Segui le procedure guidate di configurazione
    - Imposta le opzioni base di ogni plugin
4. **Crea la struttura del sito:**
    - Definisci le pagine principali (Home, Chi siamo, Contatti, ecc.)
    - Imposta il menu di navigazione
5. **Configura le impostazioni di sicurezza e backup:**
    - Attiva la protezione con Wordfence
    - Configura i backup automatici con UpdraftPlus

Questa guida ti offre i passaggi essenziali per iniziare a sviluppare un sito WordPress in locale. Ricorda che la scelta dei plugin dipende molto dal tipo specifico di sito che intendi

creare; quindi, definisci chiaramente gli obiettivi del tuo progetto prima di iniziare a installare e configurare i plugin.

Introduzione alla programmazione CSS in Wordpress

Quando si lavora con WordPress e vuoi personalizzare l'aspetto del sito attraverso il CSS (Cascading Style Sheets), ci sono diverse opzioni a disposizione. Ecco una guida completa su come iniziare con il CSS in WordPress:

1. Comprendere la gerarchia dei temi WordPress

Prima di tutto, è importante capire che WordPress utilizza una struttura gerarchica di temi:

- Il tema principale contiene i file di base
- Un eventuale tema figlio (child theme) può sovrascrivere e personalizzare il tema principale

2. Metodi per aggiungere CSS personalizzato in WordPress

Metodo 1: Usando il Personalizzatore di WordPress

Questo è il metodo più semplice per i principianti:

1. Dal pannello di amministrazione, vai a **Aspetto → Personalizza**
2. Cerca l'opzione **CSS aggiuntivo** o **CSS personalizzato** (la posizione esatta può variare in base al tema)
3. Aggiungi il tuo codice CSS nella casella di testo
4. Clicca su **Pubblica** per salvare le modifiche

Esempio di codice CSS che si potrebbe aggiungere:

```
.site-title {
    color: #ff0000; /* Cambia il colore del titolo del sito in rosso */
    font-size: 36px; /* Aumenta la dimensione del testo */
```


by: Ulises Gabriel Miranda – mail@ulises.it

```
        }
    .entry-content p {
        line-height: 1.8; /* Aumenta la spaziatura tra le righe dei paragrafi */
        }
```

Metodo 2: Creare un tema figlio (consigliato)

Questo metodo è più avanzato ma è la pratica migliore:

1. **Crea una cartella del tema figlio** nella directory wp-content/themes/
2. **Crea un file style.css** nella cartella del tema figlio con questo header:

```
/*
Theme Name: Nome Tema Padre Child
Theme URI: http://esempio.com/
Description: Tema figlio per il tema Nome Tema Padre
Author: Il tuo nome
Author URI: http://iltuosito.com/
Template: nome-tema-padre
Version: 1.0
*/
/* Importa gli stili del tema principale */
@import url("../nome-tema-padre/style.css");

/* Aggiungi qui le tue personalizzazioni CSS */
```

3. **Crea un file functions.php per caricare correttamente gli stili**:

```
<?php
function tema_figlio_enqueue_styles() {
  wp-enqueue-style('parent-style', get_template_directory_uri() . '/style.css');
```


by: Ulises Gabriel Miranda – mail@ulises.it

```
    wp-enqueue-style('child-style', get_stylesheet_uri(), array('parent-style'));
}
add_action('wp_enqueue_scripts', 'tema_figlio_enqueue_styles');
?>
```

4. **Attiva il tema figlio** in Aspetto → Temi

Metodo 3: Usando un plugin CSS personalizzato

Se preferisci non creare un tema figlio, puoi utilizzare plugin come:

- **Simple Custom CSS**
- **Custom CSS Pro**
- **WP Add Custom CSS**

Installazione tipica:

1. Vai a **Plugin → Aggiungi nuovo**
2. Cerca il plugin (es. "Simple Custom CSS")
3. Installa e attiva
4. Vai alle impostazioni del plugin per aggiungere il tuo CSS

3. Ispezionare gli elementi per scrivere CSS efficace

Per modificare elementi specifici:

1. **Usa l'ispettore del browser**:
    - Fai clic destro sull'elemento che vuoi modificare
    - Seleziona "Ispeziona" o "Ispeziona elemento"
    - Nel pannello che si apre, puoi vedere:
        - La struttura HTML

- Le classi e gli ID applicati all'elemento
- Il CSS attualmente applicato

2. **Identifica i selettori giusti**:
   - Cerca classi o ID nel codice HTML dell'elemento
   - Usa questi selettori nel tuo CSS personalizzato

Esempio pratico:

```
/* Se l'elemento ha una classe chiamata "post-title" */
.post-title {
  font-family: 'Georgia', serif;
  color: #333366;
}

/* Se l'elemento ha un ID chiamato "main-navigation" */
#main-navigation {
  background-color: #f5f5f5;
  border-bottom: 2px solid #ddd;
}
```

4. Regole CSS comuni per personalizzare WordPress

Ecco alcuni esempi di personalizzazioni frequenti:

Modifica del header:

```
.site-header {
  background-color: #2c3e50;
  padding: 20px 0;
```

```
}

.site-title a {
  color: #ffffff;
  font-weight: bold;
}
```

Personalizzazione dei pulsanti:

```
.button, .btn, .wp-block-button__link {
  background-color: #3498db;
  color: white;
  padding: 10px 20px;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}

.button:hover, .btn:hover, .wp-block-button__link:hover {
  background-color: #2980b9;
}
```

Modifica dello stile dei post:

```
.entry-title {
  font-size: 28px;
  border-bottom: 1px solid #eee;
  padding-bottom: 10px;
}
```

```
.entry-meta {
  font-size: 14px;
  color: #777;
}

.entry-content {
  font-size: 16px;
  line-height: 1.6;
}
```

5. CSS Responsive per diverse dimensioni di schermo

È importante che i contenuti funzionino bene su tutti i dispositivi (oggigiorno oltre il 90% degli accessi ad ogni sito web va eseguito da dispositivi mobile):

```
/* Stile di base per desktop */
.contenuto-principale {
  padding: 30px;
  margin: 0 auto;
  max-width: 1200px;
}

/* Media query per tablet */
@media screen and (max-width: 768px) {
  .contenuto-principale {
    padding: 20px;
    max-width: 100%;
```


by: Ulises Gabriel Miranda – mail@ulises.it

```
    }

    .site-title {
      font-size: 24px;
    }
  }

  /* Media query per smartphone */
  @media screen and (max-width: 480px) {
    .contenuto-principale {
      padding: 10px;
    }

    .entry-title {
      font-size: 22px;
    }
  }
```

6. Suggerimenti avanzati

1. **Usa l'ispettore per testare le modifiche in tempo reale**:
    - Nel pannello Stili dell'ispettore, puoi modificare il CSS e vedere immediatamente i risultati
    - Poi copia le modifiche funzionanti nel file CSS

2. **Presta attenzione alla specificità dei selettori**:
    - A volte hai bisogno di selettori più specifici per sovrascrivere gli stili esistenti

- Esempio: *.content .entry-content p* è più specifico di solo *p*

3. **Usa i prefissi del fornitore per le proprietà CSS più recenti:**

```
.elemento {
  -webkit-transition: all 0.3s ease;
  -moz-transition: all 0.3s ease;
  -o-transition: all 0.3s ease;
  transition: all 0.3s ease;
}
```

4. **Organizza il contenuto CSS in sezioni:**

```
/* =Header
---------------------------------------------- */
/* Stili per l'header qui */

/* =Navigation
---------------------------------------------- */
/* Stili per il menu qui */

/* =Content
---------------------------------------------- */
/* Stili per il contenuto qui */
```

Consiglio:

Inizia con piccole modifiche e testale prima di procedere con altre. Ricorda sempre di fare un backup del sito prima di apportare modifiche significative (si può usare lo staging) e, se possibile, testare le modifiche in un ambiente di sviluppo locale prima di applicarle al sito live. Le installazioni locali d Wordpress consentono anche di testare ogni impostazione.