# Il modello web standard

- Il modello web standard si basa su HTML, CSS e JavaScript.
- HTML definisce la struttura della pagina.
- CSS gestisce la presentazione e lo stile.
- JavaScript aggiunge interattività e dinamismo.

# HTML, CSS e JavaScript

- **HTML**: linguaggio di markup per la struttura della pagina.
- **CSS**: fogli di stile per il design e l'impaginazione.
- **JavaScript**: linguaggio di scripting per il comportamento dinamico.

# Cos'è l'HTML

- HyperText Markup Language.
- Struttura il contenuto di una pagina web.
- Utilizza **tag** per definire gli elementi.

```
<!DOCTYPE html>
<html>
<head>
  <title>Pagina di esempio</title>
</head>
<body>
  <h1>Ciao, mondo!</h1>
</body>
</html>
```

# Gli elementi, i tag e gli attributi

- **Elemento**: una combinazione di tag di apertura, contenuto e tag di chiusura.
- **Tag**: definiscono l'inizio e la fine di un elemento.
- **Attributi**: forniscono informazioni aggiuntive sugli elementi.

Esempio di attributo:

```
<img src="immagine.jpg" alt="Descrizione dell'immagine">
```

# Il documento: doctype e parti principali

- `<!DOCTYPE html>` : definisce la versione HTML.
- `<html>` : radice del documento.
- `<head>` : metadati, collegamenti a CSS e JavaScript.
- `<body>` : contenuto visibile della pagina.

# La formattazione del codice

- Indentazione corretta per leggibilità.
- Uso coerente di maiuscole/minuscole nei tag.
- Commenti per spiegare il codice:

```
<!-- Questo è un commento HTML -->
```

# I principali elementi semantici introdotti da HTML5

- `<header>` : intestazione della pagina o sezione.
- `<nav>` : navigazione principale.
- `<section>` : sezione autonoma.
- `<article>` : contenuto indipendente.
- `<footer>` : piè di pagina.

## Esempio:

```
<header>
  <h1>Benvenuti</h1>
</header>
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Contatti</a></li>
  </ul>
</nav>
```

# Gli elementi principali di HTML

- Strutturali: `<div>`, `<span>`.
- Testuali: `<h1>` - `<h6>`, `<p>`, `<strong>`, `<em>`.
- Liste: `<ul>`, `<ol>`, `<li>`.
- Tabelle: `<table>`, `<tr>`, `<td>`, `<th>`.
- Form: `<form>`, `<input>`, `<label>`, `<select>`, `<textarea>`.

# Eventi in JavaScript

- **Eventi**: azioni rilevate dal browser (click, scroll, input, ecc.).
- Ascoltatori di eventi con `addEventListener`:

```
document.getElementById("bottone").addEventListener("click", function() {
  alert("Bottone cliccato!");
});
```

- Eventi comuni:
  - `click`
  - `mouseover`
  - `keydown`
  - `submit`

# Arrays in JavaScript

- Strutture dati per collezioni ordinate.
- Dichiarazione:

```
let numeri = [1, 2, 3, 4, 5];
```

- Metodi comuni:
  - `push()`, `pop()`, `shift()`, `unshift()`
  - `map()`, `filter()`, `reduce()`

Esempio:

```
let numeriDoppi = numeri.map(num => num * 2);
console.log(numeriDoppi);
```

# Promises in JavaScript

- Meccanismo per la gestione di operazioni asincrone.
- Stati di una Promise:
  - **Pending**: in attesa di completamento.
  - **Resolved (Fulfilled)**: operazione completata con successo.
  - **Rejected**: operazione fallita.

Esempio:

```
let promessa = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Dati ricevuti!"), 2000);
});

promessa.then((messaggio) => console.log(messaggio));
```

# Promises in JavaScript

- **Async/Await** per una gestione più pulita:

```
async function getData() {
  let response = await promessa;
  console.log(response);
}
getData();
```

# Operatore Spread in JavaScript

- L'operatore `...` consente di espandere iterabili (array, oggetti, stringhe).
- **Espansione di array**:

```
let numeri = [1, 2, 3];
let altriNumeri = [...numeri, 4, 5, 6];
console.log(altriNumeri);
```

- **Clonazione di oggetti**:

```
let obj1 = { a: 1, b: 2 };
let obj2 = { ...obj1, c: 3 };
console.log(obj2);
```

- **Passaggio di argomenti**:

```
function somma(a, b, c) {
```

# AJAX (Asynchronous JavaScript and XML)

- Comunicazione asincrona con il server.
- Utilizzo di `XMLHttpRequest` o `fetch()`:

```
fetch("https://jsonplaceholder.typicode.com/todos/1")
  .then(response => response.json())
  .then(data => console.log(data));
```

- Applicazione in chiamate API per aggiornare la pagina senza ricaricare.

# Introduzione a jQuery

- Libreria JavaScript per facilitare l'interazione con il DOM.
- Selezione elementi:

```
$("#mioID").hide();
```

- Gestione eventi:

```
$("button").click(function() {   alert("Bottone cliccato!"); });
```

- Richieste AJAX semplificate:

```
$.get("https://jsonplaceholder.typicode.com/todos/1", function(data) {
  console.log(data);
});
```

# Conclusioni

- HTML fornisce la struttura.
- CSS definisce lo stile.
- JavaScript aggiunge interattività.
- jQuery semplifica le operazioni sul DOM.
- AJAX consente aggiornamenti dinamici della pagina.

Grazie per l'attenzione! 🚀