JavaScript e TypeScript: Fondamenti per lo Sviluppatore Full Stack

JavaScript Essenziale

Concetti Fondamentali

- **Tipi di dati**: `string`, `number`, `boolean`, `null`, `undefined`, `object`, `symbol`, `bigint`
- **Variabili**: `var`, `let`, `const` e le loro differenze (scope, hoisting)
- **Operatori**: aritmetici, logici, di confronto, ternari
- **Strutture di controllo**: `if/else`, `switch`, cicli (`for`, `while`, `do-while`)

Funzioni

```javascript
// Dichiarazione funzione
function nomeFunzione(param1, param2) {
  return param1 + param2;
}

// Function expression
const nomeFunzione = function(param1, param2) {
  return param1 + param2;
};

// Arrow function
const nomeFunzione = (param1, param2) => param1 + param2;
```

Array e Oggetti

javascript

```javascript
// Array methods da padroneggiare
const array = [1, 2, 3, 4, 5];
array.map(x => x * 2);        // [2, 4, 6, 8, 10]
array.filter(x => x % 2 === 0); // [2, 4]
array.reduce((acc, curr) => acc + curr, 0); // 15
array.find(x => x > 3);       // 4
array.some(x => x > 3);       // true
array.every(x => x > 0);      // true

// Oggetti e manipolazione
const obj = { nome: 'Mario', età: 30 };
const { nome, età } = obj;    // Destructuring
const newObj = { ...obj, professione: 'Developer' }; // Spread
```

Async JavaScript

javascript

```javascript
// Promises
const promise = new Promise((resolve, reject) => {
  // Operazione asincrona
  if (success) {
    resolve(data);
  } else {
    reject(error);
  }
});

promise
  .then(data => console.log(data))
  .catch(error => console.error(error));

// Async/Await
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error:', error);
  }
}
```

Pattern JS Avanzati

- **Closure**: Funzioni che mantengono accesso allo scope in cui sono state create
- **IIFE** (Immediately Invoked Function Expression): `(function() { /* code */ })();`
- **Module Pattern**: Organizzazione di codice incapsulato
- **Prototypal Inheritance**: Meccanismo di ereditarietà di JavaScript

TypeScript Fondamentale

Tipizzazione di Base

typescript

```typescript
// Tipi fondamentali
let nome: string = 'Mario';
let età: number = 30;
let isActive: boolean = true;
let skills: string[] = ['JavaScript', 'TypeScript', 'React'];
let tuple: [string, number] = ['Mario', 30];
```

Interfacce e Tipi

typescript

```typescript
// Interface
interface User {
  id: number;
  name: string;
  email: string;
  active?: boolean; // Proprietà opzionale
  readonly createdAt: Date; // Proprietà in sola lettura
}

// Type
type UserRole = 'admin' | 'user' | 'guest';
type User = {
  id: number;
  name: string;
  role: UserRole;
};
```

Generics

typescript

```typescript
// Funzione generica
function identity<T>(arg: T): T {
  return arg;
}

// Classe generica
class Box<T> {
  private content: T;

  constructor(value: T) {
    this.content = value;
  }

  getValue(): T {
    return this.content;
  }
}

// Interfaccia generica
interface Repository<T> {
  getById(id: string): T;
  getAll(): T[];
  create(item: T): void;
}
```

TypeScript e React

```typescript
// Props tipizzate
interface ButtonProps {
  text: string;
  onClick: () => void;
  variant?: 'primary' | 'secondary';
}

const Button: React.FC<ButtonProps> = ({ text, onClick, variant = 'primary' }) => {
  return (
    <button
      className={`btn btn-${variant}`}
      onClick={onClick}
    >
      {text}
    </button>
  );
};

// Hook tipizzati
const [user, setUser] = useState<User | null>(null);
```

Esercizi Pratici

JavaScript

1. Implementa una funzione che filtra e trasforma un array di oggetti
2. Crea una catena di promise per operazioni sequenziali
3. Implementa un sistema di caching utilizzando closure

TypeScript

1. Definisci interfacce per un sistema di e-commerce (Prodotto, Utente, Ordine)
2. Crea una funzione generica per la manipolazione sicura dei dati
3. Implementa un componente React tipizzato con props e state complessi

Risorse per l'Approfondimento

- MDN Web Docs
- TypeScript Handbook
- JavaScript.info
- TypeScript Deep Dive