JavaScript Scope, Hoisting, Closure e altro

Abbiamo analizzato i concetti fondamentali di JavaScript attraverso esempi reali. Di seguito una sintesi strutturata e dettagliata della conversazione.

1. Scope

Lo scope e il contesto in cui una variabile e visibile:

- Variabili dichiarate fuori da una funzione Scope GLOBALE
- Variabili dichiarate dentro una funzione Scope LOCALE
- Con `var`, lo scope e sempre di funzione
- Con `let` e `const`, lo scope e di blocco `{}`

Esempio:

```javascript
var x = 1;
function test() {
  var x = 2;
  console.log(x); // 2
}
test();
console.log(x); // 1
```

2. Hoisting

JavaScript solleva le dichiarazioni di variabili (e funzioni):

- `var` viene hoistata con valore `undefined`
- `let` e `const` vengono hoistate MA generate in uno stato chiamato TDZ (Temporal Dead Zone)

Esempio:

```javascript
function saluta() {
  console.log(b); // undefined
  var b = 1;
}
```

Interpretabile come:

```javascript
function saluta() {
  var b;
  console.log(b);
  b = 1;
}
```

3. Shadowing

Una variabile locale puo oscurare una variabile globale con lo stesso nome.

Esempio:
```javascript
var b = "globale";
function esempio() {
  var b = "locale";
  console.log(b); // stampa "locale"
}
```

4. var vs let/const e Temporal Dead Zone (TDZ)

Differenze principali:

- `var`: hoistata e accessibile come `undefined`
- `let` e `const`: hoistate ma NON accessibili prima della dichiarazione (TDZ)

Esempio:
```javascript
{
  console.log(a); // ReferenceError
  let a = 10;
}
```

5. Function Declaration vs Function Expression

Function Declaration:
```javascript
saluta(); // ok
function saluta() {
  console.log("ciao");
}
```

Function Expression:
```javascript
saluta(); // TypeError
var saluta = function() {
  console.log("ciao");
};
```

6. Closures

Una funzione interna puo ricordare le variabili dello scope esterno anche dopo che questultimo e terminato.

Esempio:
```javascript
function creaContatore() {
  let contatore = 0;
  return function() {
```

```
    contatore++;
    console.log(contatore);
  };
}

const conta = creaContatore();
conta(); // 1
conta(); // 2
conta(); // 3
```

Spiegazione passo passo:

- `creaContatore` crea una variabile `contatore` e restituisce una funzione
- La funzione restituita ricorda `contatore` grazie alla closure
- Ogni volta che chiami `conta()`, usi lo stesso `contatore` chiuso

Esempio ulteriore:
```javascript
const c1 = creaContatore();
const c2 = creaContatore();
c1(); // 1
c1(); // 2
c2(); // 1
```

Conclusione

Abbiamo esplorato:
1. Scope
2. Hoisting
3. Shadowing
4. var/let/const e TDZ
5. Function declaration vs expression
6. Closures

Ogni concetto e stato supportato da esempi pratici, spiegazioni dettagliate e differenze critiche.