Tag 1: Einführung in die Programmierung und JavaScript

1. Computer Programming Grundlagen (ca. 2 Stunden)

1.1 Einführung in Programmierbegriffe

- **Interpreter vs. Compiler:**
  - Ein **Interpreter** ist ein Programm, das Quellcode Zeile für Zeile ausführt. Jede Zeile wird interpretiert, während das Programm läuft. Dies bedeutet, dass der Interpreter den Quellcode liest, analysiert und dann sofort ausführt. JavaScript ist ein Beispiel für eine Sprache, die interpretiert wird. Der Browser, in dem der Code ausgeführt wird, enthält den JavaScript-Interpreter.
  - Ein **Compiler** übersetzt den gesamten Quellcode in Maschinensprache, bevor das Programm ausgeführt wird. Dies führt oft zu einer schnelleren Ausführung, da der Code nicht bei jeder Ausführung analysiert werden muss. C und C++ sind typische Beispiele für kompilierte Sprachen.

  Beispiele und Erklärung:
  - Wenn du ein JavaScript-Programm in der Konsole eines Browsers schreibst und sofort ausführst, wird es Zeile für Zeile interpretiert.
  - Ein C++-Programm muss jedoch zuerst kompiliert werden, bevor es ausgeführt werden kann.

  Übung:
  - Öffne die Konsole deines Browsers und gib den folgenden Code ein:

    ```
    console.log("Hallo, dies ist mein erstes JavaScript-Programm!");
    ```

    Erläutere deinen Schülern, dass dieser Code Zeile für Zeile interpretiert wird, wenn sie auf "Enter" drücken.

- **Client-side vs. Server-side Programmierung:**
  - **Client-side**-Programmierung bedeutet, dass der Code auf dem Computer des Benutzers (also im Browser) ausgeführt wird. Dies macht den Code schneller und reaktionsfähiger, da keine Anfrage an einen Server gesendet werden muss, um die Ergebnisse zu erhalten. JavaScript ist das typische Beispiel für eine clientseitige Programmiersprache.
  - **Server-side**-Programmierung bedeutet, dass der Code auf einem Server ausgeführt wird, und die Ergebnisse werden dann an den Browser des Benutzers gesendet. Sprachen wie PHP oder Node.js sind häufige Beispiele für serverseitige Sprachen.

  Beispiel:
  - Wenn du eine Schaltfläche auf einer Webseite drückst und JavaScript den Text auf der Seite sofort ändert, ohne die Seite neu zu laden, handelt es sich um Client-side Code.
  - Wenn du ein Formular sendest und die Daten an einen Server gesendet werden, um verarbeitet zu werden, handelt es sich um serverseitige Programmierung.

  Übung:
  - Erstelle eine einfache HTML-Seite, die JavaScript verwendet, um den Text zu ändern, wenn eine Schaltfläche gedrückt wird. Lass die Schüler verstehen, dass dies clientseitig ausgeführt wird.

    ```
    <!DOCTYPE html>
    <html>
    <head>
        <title>Client-side Beispiel</title>
    </head>
    <body>
        <h1 id="title">Drücke die Schaltfläche!</h1>
        <button onclick="changeText()">Ändere den Text</button>

        <script>
            function changeText() {
                document.getElementById("title").innerHTML = "Der Text wurde geändert!";
            }
        </script>
    </body>
    </html>
    ```

1.2 Einrichten der Entwicklungsumgebung

- **Grundlegende Tools für die JavaScript-Entwicklung:**
  - **Code-Editor:** Ein Texteditor, der für die Webentwicklung geeignet ist. Ein Beispiel ist **Visual Studio Code**, der viele nützliche Funktionen bietet, wie Syntaxhervorhebung, Autovervollständigung und integrierte Debugging-Tools.
  - **Konsole:** Alle modernen Browser haben Entwicklerwerkzeuge, die eine **Konsole** bereitstellen, in der du JavaScript-Code direkt ausführen und testen kannst.
  - **Debugger:** Die Entwicklerwerkzeuge enthalten auch einen **Debugger**, der verwendet wird, um Schritt für Schritt durch den Code zu gehen, Breakpoints zu setzen und den aktuellen Zustand des Programms zu untersuchen.

  Übung:
  - Lasse deine Schüler Visual Studio Code oder einen ähnlichen Editor installieren.

- Zeige, wie man den Browser öffnet und auf die Entwicklerwerkzeuge zugreift (meistens über F12 oder Rechtsklick -> "Untersuchen").

1.3 Client-side Ausführung von JavaScript

- **JavaScript in HTML einbetten:**
  - JavaScript kann direkt in HTML eingebettet werden, indem man das `<script>`-Tag verwendet. Es kann sowohl im `<head>` - als auch im `<body>`-Tag platziert werden.
  - Alternativ kann JavaScript in einer separaten Datei gespeichert werden, die mit dem `src`-Attribut des `<script>`-Tags eingebunden wird.

  Beispiel:

  - **Interne Einbindung:**

    ```
    <script>
        console.log("Interner JavaScript-Code");
    </script>
    ```

  - **Externe Einbindung:**

    ```
    <script src="script.js"></script>
    ```

    In der Datei **script.js**:

    ```
    console.log("Externer JavaScript-Code");
    ```

2. Einführung in HTML und JavaScript (ca. 2 Stunden)

2.1 HTML-Grundlagen: Wie wird JavaScript eingebettet?

- HTML ist die Struktur der Webseite. Alle visuellen Elemente auf einer Webseite wie Text, Bilder, Schaltflächen usw. werden durch HTML definiert. JavaScript ermöglicht es, diese Elemente dynamisch zu ändern.

  Grundlegendes HTML-Template:

  ```
  <!DOCTYPE html>
  <html>
  <head>
      <title>Meine erste HTML-Seite</title>
  </head>
  <body>
      <h1>Willkommen!</h1>
      <p>Dies ist meine erste Seite.</p>
  </body>
  </html>
  ```

2.2 Erste Schritte mit JavaScript in HTML

- JavaScript kann verwendet werden, um die Inhalte und das Verhalten der Seite zu ändern.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Beispiel</title>
</head>
<body>
    <p id="text">Dieser Text wird geändert!</p>
    <button onclick="changeText()">Ändere den Text</button>

    <script>
        function changeText() {
            document.getElementById("text").innerHTML = "Der Text wurde geändert!";
        }
    </script>
</body>
</html>
```

2.3 JavaScript in der Konsoloe ausführen

- Die Konsole ist ein wesentliches Werkzeug, um Code direkt zu testen und Fehler zu debuggen. Schüler sollten lernen, wie man einfache Befehle eingibt und die Ausgabe sieht.

```
let name = "Max";
console.log(name);
```

Variablen, Datentypen und Typumwandlung

1. Variablen und Datenarten

1.1 Variablen deklarieren und initialisieren

- JavaScript bietet drei Möglichkeiten, Variablen zu deklarieren: var, let und const.
    - `var` : Wird verwendet, um Variablen zu deklarieren, die im gesamten Funktions- oder globalen Scope verfügbar sind. Es ist alt und wird nicht mehr empfohlen, da es zu unbeabsichtigten Problemen führen kann.
    - `let` : Wird verwendet, um block-skopierte Variablen zu deklarieren. Es wird für variablen verwendet, die sich ändern können.
    - `const` : Wird verwendet, um Konstanten zu deklarieren, deren Wert nicht geändert werden kann.

```
var age = 25;
console.log(age); // Ausgabe: 25

let name = "Sarah";
console.log(name); // Ausgabe: Sarah

const pi = 3.14159;
console.log(pi); // Ausgabe: 3.14159
```

1.2 Primitive Datentypen

- JavaScript verfügt über sechs primitive Datentypen:
    - `boolean` : Ein Wahrheitswert (true oder false).
    - `number` : Eine Zahl, entweder Ganzzahl oder Gleitkommazahl.
    - `bigint` : Eine große Ganzzahl (größer als Number.MAX_SAFE_INTEGER).
    - `undefined` : Eine Variable, die deklariert, aber nicht initialisiert wurde, hat den Wert undefined.
    - `null` : Repräsentiert einen leeren oder nicht existierenden Wert.
    - `string` : Eine Zeichenkette, die Text enthält.

```
let isLoggedIn = true;
let hasAccess = false;
console.log(isLoggedIn, hasAccess); // Ausgabe: true false

// Number
let distance = 42.195; // Marathon-Distanz in Kilometern
console.log(distance); // Ausgabe: 42.195

// BigInt
let bigNumber = 9007199254740991n;
console.log(bigNumber); // Ausgabe: 9007199254740991n

// Undefined
let name;
console.log(name); // Ausgabe: undefined

// Null
let result = null;
console.log(result); // Ausgabe: null

// String
let greeting = "Hallo, Welt!";
console.log(greeting); // Ausgabe: Hallo, Welt!
```