Erläuterung zum HTML-Code des IT-Projektes Webentwicklung

HTML-Dokumentstruktur

```
<!DOCTYPE html>
```

- **Erläuterung**: Diese Zeile gibt an, dass es sich um ein HTML5-Dokument handelt. Sie ist erforderlich, um den Browser darüber zu informieren, dass das Dokument in HTML5 geschrieben ist.

```
<html lang="de">
```

- **Erläuterung**: Das `<html>`-Tag ist das Wurzel-Element des HTML-Dokuments. Das Attribut `lang="de"` gibt an, dass die Sprache des Dokuments Deutsch ist. Dies ist wichtig für Suchmaschinen und Screenreader, um den Inhalt korrekt zu interpretieren.

Kopfbereich des Dokuments

```
<head>
    <title>HTML-Projekt</title>
    <meta charset="UTF-8">
    <meta name="description" content="Darstellung unseres Projektes">
    <meta name="keywords" content="HTML5, CSS3, IT-Projekt, Informatik, FOS">
    <meta name="Herold" content="Webdesign">
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

- **`<head>`**: Dieser Abschnitt enthält Metainformationen über das Dokument, die nicht direkt auf der Seite angezeigt werden.
- **`<title>`**: Der Titel des Dokuments, der im Browser-Tab angezeigt wird. In diesem Fall "HTML-Projekt".
- **`<meta charset="UTF-8">`**: Legt die Zeichencodierung des Dokuments auf UTF-8 fest, was eine breite Palette von Zeichen unterstützt, einschließlich Sonderzeichen.
- **`<meta name="description">`**: Eine kurze Beschreibung des Inhalts der Seite, die von Suchmaschinen verwendet wird.
- **`<meta name="keywords">`**: Eine Liste von Schlüsselwörtern, die den Inhalt der Seite beschreiben. Diese werden ebenfalls von Suchmaschinen verwendet, sind aber heutzutage weniger wichtig.
- **`<meta name="author">`**: Gibt den Autor des Dokuments an.
- **`<link rel="stylesheet" href="style.css">`**: Verknüpft die externe CSS-Datei `style.css` mit dem HTML-Dokument, um das Styling der Seite zu steuern.
- **`<meta name="viewport">`**: Diese Zeile sorgt dafür, dass die Seite auf mobilen Geräten korrekt skaliert wird. `width=device-width` bedeutet, dass die Breite des Viewports der Breite des Geräts entspricht, und `initial-scale=1.0` setzt den anfänglichen Zoomfaktor.

Hauptinhalt des Dokuments

```
<body>
    <header>
        <img src="bilder/Schullogo.png" alt="Logo">
        <h1>HTML5 und CSS3</h1>
        <h2>Projekt Webentwicklung im Fach Informatik</h2>
    </header>
```

- **`<body>`**: Der Hauptinhalt des HTML-Dokuments. Alles, was im Browser angezeigt wird, befindet sich innerhalb dieses Tags.
- **`<header>`**: Ein semantisches Element, das den Kopfbereich der Seite enthält. Hier sind das Logo, der Haupttitel (`<h1>`) und der Untertitel (`<h2>`) untergebracht.
- **`<img src="bilder/Schullogo.png" alt="Logo">`**: Ein Bild-Tag, das das Logo anzeigt. Das `src`-Attribut gibt den Pfad zum Bild an, und das `alt`-Attribut beschreibt das Bild für Screenreader und zeigt den Text an, falls das Bild nicht geladen werden kann.
- **`<h1>` und `<h2>`**: Überschriften, die den Titel und den Untertitel des Projekts darstellen. Diese sind wichtig für die Strukturierung des Inhalts und die SEO-Optimierung.

```
<nav>
    <ul>
        <li><a href="index.html">Startseite</a></li>
        <li><a href="unterseite1.html">Vorgehen</a></li>
        <li><a href="unterseite2.html">HTML/CSS-Befehle</a></li>
        <li><a href="unterseite3.html">Freiraum</a></li>
        <li><a href="kontakt.html">Kontakt</a></li>
    </ul>
</nav>
```

- **`<nav>`**: Ein semantisches Element, das die Navigation der Seite enthält.
- **`<ul>`**: Eine ungeordnete Liste, die die Navigationslinks enthält.
- **`<li>`**: Listenelemente, die jeweils einen Link zur entsprechenden Seite enthalten.
- **`<a href="...">`**: Ein Anker-Tag, das einen Hyperlink zu einer anderen Seite oder Ressource darstellt.

```
<main>
    <article>
        <h2>Unser IT-Projekt</h2>
        <p>
            In unserem IT-Projekt <strong>Webentwicklung & Webdesign</strong> lernen
wir die wichtigsten Grundlagen von HTML5 und CSS3.
        </p>
        <p>
            Gestalten Sie eine <strong>eigene Webpräsenz</strong>.
            Sie dürfen und sollen diese Vorlage frei verändern und umgestalten.
            Integrieren Sie eigene Texte, Bilder, Links, Dokumente, usw. in Ihre selbst
erstellte Webpräsenz.
            Gestalten Sie Ihr eigenes Design - nutzen Sie CSS!
        </p>
    </article>
    <aside>
        <figure>
            <img src="bilder/teamwork.jpg" alt="Teamwork">
            <figcaption>Für unser IT-Projekt arbeiten wir in Gruppen</figcaption>
        </figure>
        <img src="bilder/recht.jpg" alt="Urheberrecht">
    </aside>
</main>
```

- **`<main>`**: Ein semantisches Element, das den Hauptinhalt der Seite umfasst. Es sollte den Hauptinhalt der Seite enthalten, der für das Verständnis der Seite wichtig ist.
- **`<article>`**: Ein semantisches Element, das einen unabhängigen Inhalt darstellt. Hier wird das Thema des IT-Projekts beschrieben.
- **`<h2>`**: Eine zweite Überschrift, die den Titel des Artikels angibt.
- **`<p>`**: Absatz-Tags, die den Textinhalt des Artikels enthalten. Die Verwendung von `<strong>` hebt bestimmte Wörter hervor.
- **`<aside>`**: Ein semantisches Element, das verwandte Inhalte oder Informationen enthält, die nicht zum Hauptinhalt gehören.
- **`<figure>`**: Ein Element, das ein Bild und eine zugehörige Bildunterschrift (`<figcaption>`) enthält. Dies ist nützlich, um den Kontext des Bildes zu erklären.

```
<footer>
    <p>FOSOBOS Bamberg - Ohmstr. 17 - herold@fos-bamberg.de</p>
</footer>
```

- **`<footer>`**: Ein semantisches Element, das den Fußbereich der Seite darstellt. Es enthält in der Regel Informationen über die Seite, wie Urheberrechtshinweise oder Kontaktinformationen.
- **`<p>`**: Ein Absatz, der die Kontaktdaten der Institution angibt.

Erläuterung zum CSS-Code des IT-Projektes Webentwicklung

1. CSS-Variablen

```
:root {
    --primärfarbe: #fbce19;
    --sekundärfarbe: #4a79b0;
    --textfarbe: black;
    --footertextfarbe: #4a79b0;
}
```

Erläuterung: CSS-Variablen ermöglichen es, Werte zu speichern, die an mehreren Stellen im CSS verwendet werden können. Sie werden im `:root`-Selektor definiert, sodass sie im gesamten Dokument verfügbar sind.

---

2. Universeller Selektor

```
* {
    box-sizing: border-box;
    padding: 0;
    margin: 0;
    word-wrap: break-word;
}
```

Erläuterung: Der universelle Selektor `*` wendet die angegebenen Stile auf alle Elemente im Dokument an. Hier wird das Box-Modell mit `box-sizing: border-box` angepasst, sodass Padding und Border in der Gesamtbreite und -höhe eines Elements enthalten sind. Padding und Margin werden auf 0 gesetzt, um einen einheitlichen Abstand zu gewährleisten.

---

3. Bildstile

```
img {
    max-width: 100%;
    height: auto;
}
```

Erläuterung: Diese Regeln sorgen dafür, dass Bilder responsiv sind. `max-width: 100%` stellt sicher, dass Bilder die Breite ihres Containers nicht überschreiten, während `height: auto` das Seitenverhältnis beibehält.

---

4. Body-Stile

```
body {
    background-color: white;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 1rem;
    max-width: 75rem;
    margin: 0 auto;
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

Erläuterung: Hier wird der Hintergrund des Bodys auf weiß gesetzt, und die Schriftart wird festgelegt. `max-width` begrenzt die Breite des Inhalts, und `margin: 0 auto` zentriert den Body. `display: flex` aktiviert Flexbox, um die Anordnung der Kinder zu steuern.

---

5. Header-Stile

```
header {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 100%;
    max-width: 75rem;
    background-color: white;
    border: 0.625em solid var(--primärfarbe);
    border-radius: 1.5em;
    margin-top: 1.25em;
    margin-bottom: 1.25em;
    padding: 0.625em;
    overflow: auto;
}
```

Erläuterung: Der Header verwendet Flexbox, um die Anordnung seiner Kinder (Logo, Titel, Untertitel) zu steuern. Die Breite wird auf 100% gesetzt, und es wird eine Border sowie abgerundete Ecken hinzugefügt.

---

6. Navigation

```
nav {
    padding: 0.625em;
}
```

Erläuterung: Hier wird ein Innenabstand für das Navigationsmenü festgelegt, um sicherzustellen, dass der Inhalt nicht direkt am Rand des Containers sitzt.

---

7. Artikel- und Aside-Stile

```
article {
    padding: 1.25em;
    color: var(--textfarbe);
    line-height: 1.5;
}

aside {
    padding: 0.625em;
    border-radius: 1.5em;
    background-color: var(--
sekundärfarbe);
    color: white;
}
```

Erläuterung: Das `article` erhält einen Innenabstand und eine Textfarbe, während das `aside` eine Hintergrundfarbe, abgerundete Ecken und eine weiße Schriftfarbe hat.

---

8. Footer-Stile

```
footer {
    width: 100%;
    max-width: 75rem;
    background-color: white;
    border: 0.625em solid var(--
primärfarbe);
    border-radius: 1.5em;
    min-height: 5em;
    text-align: center;
    color: var(--footertextfarbe);
    margin-top: 1.25em;
}
```

Erläuterung: Der Footer nimmt die volle Breite ein, hat eine maximale Breite und eine Hintergrundfarbe. Er ist zentriert und hat eine minimale Höhe.

9. Medienabfragen und Grid-Layout

```
@media only screen and (min-width:
768px) {
    body {
        display: grid;
        grid-template-areas:
            "header header header"
            "nav main main"
            "footer footer footer";
        grid-template-columns: 1fr 3fr;
        gap: 1.25em;
        max-width: 75rem;
    }

    header {
        grid-area: header;
    }

    nav {
        grid-area: nav;
        align-self: start;
    }

    main {
        grid-area: main;
        display: grid;
        grid-template-areas: "article
aside";
        grid-template-columns: 2fr 1fr;
        gap: 1.25em;
    }

    article {
        grid-area: article;
    }

    aside {
        grid-area: aside;
    }

    footer {
        grid-area: footer;
    }
}
```

Erläuterung: Medienabfragen ermöglichen es, CSS-Regeln abhängig von der Bildschirmgröße anzuwenden. In diesem Beispiel wird das Layout für Bildschirme mit einer Breite von mindestens 768 Pixeln angepasst, um ein responsives Design zu gewährleisten.

- **`display: grid;`**: Aktiviert das Grid-Layout für den `body`. Dies ermöglicht eine flexible Anordnung von Elementen in einem zweidimensionalen Raster.
- **`grid-template-areas`**: Definiert, wie die verschiedenen Bereiche der Seite angeordnet sind. Hier wird festgelegt, dass der Header über die gesamte Breite oben angezeigt wird, gefolgt von der Navigation (links) und dem Hauptinhalt (rechts). Der Footer wird ebenfalls über die gesamte Breite am Ende angezeigt.
- **`grid-template-columns: 1fr 3fr;`**: Legt die Spaltenbreiten im Grid fest. In diesem Fall nimmt die linke Spalte (Navigation) 1 Teil des verfügbaren Platzes ein, während die rechte Spalte (Hauptinhalt) 3 Teile einnimmt. Dies sorgt für ein ausgewogenes Layout, in dem der Hauptinhalt mehr Platz erhält.
- **`gap: 1.25em;`**: Definiert den Abstand zwischen den Grid-Elementen. Dies sorgt dafür, dass es einen gleichmäßigen Abstand zwischen den verschiedenen Bereichen der Seite gibt.
- **`align-self: start;`**: Diese Regel für das `nav`-Element stellt sicher, dass die Navigation am Anfang des verfügbaren Platzes ausgerichtet ist, wodurch sie immer oben bleibt.
- **`grid-area`**: Diese Eigenschaft wird für jedes Element verwendet, um zu definieren, in welchem Bereich des Grids es platziert werden soll. Zum Beispiel wird das `header`-Element im `grid-area: header` platziert, das `nav` im `grid-area: nav`, und so weiter.
- **Inneres Grid für `main`**: Innerhalb des `main`-Elements wird ein weiteres Grid aktiviert, um den `article` und `aside` nebeneinander anzuzeigen. Hier wird `grid-template-areas: "article aside";` verwendet, um die Struktur festzulegen, und `grid-template-columns: 2fr 1fr;`, um die Breite der Spalten anzupassen.
  Der `article` erhält mehr Platz als der `aside`.