Lernziele Leistungsbeurteilung 01 (LB01)

Modul: M291 Web-Frontend
Prüfungsform: Moodle-Test (Multiple Choice / Single Choice)
Dauer: 30 Minuten
Sozialform: Einzelarbeit

Die LB01 deckt sowohl das repetierte Vorwissen aus den vorbereitenden Videos und den Modulen 287 / 288 als auch die neuen Inhalte aus dem Unterricht ab.

1. UI-Analyse und HTML-Struktur

UI-Designs lesen

Sie können ein UI-Design (z. B. in Figma) in funktionale Bausteine zerlegen, indem Sie in sechs Schritten vorgehen: Grobscan, Navigation, Sections, Gruppierung, Layout und Wiederholungen erkennen. Sie ordnen die Bausteine den Klassen **Layout**, **Formular**, **Navigation** oder **Anzeige** zu.

Semantik

Sie wählen für UI-Bereiche die korrekten semantischen HTML-Tags aus (z. B. `<header>`, `<nav>`, `<main>`, `<section>`, `<footer>`) und können begründen, warum semantisches HTML Vorteile gegenüber reinen `<div>`-Strukturen bietet.

Strukturierung

Sie können Elemente sinnvoll gruppieren, sprechende Klassennamen vergeben und ein HTML-Grundgerüst mit korrekter Verlinkung von CSS (`<link>`) und JavaScript (`<script>`) aufbauen.

Bilder und srcset

Sie verstehen den Einsatz von `srcset`, um Bilder responsiv für verschiedene Bildschirmauflösungen oder Pixeldichten bereitzustellen, und können den Unterschied zum einfachen `src`-Attribut erklären.

2. CSS Layout und Styling

Schriftgrössen und Einheiten

Sie kennen den Unterschied zwischen `px`, `em` und `rem` und wissen, wann welche Einheit sinnvoll eingesetzt wird. Sie wenden `rem` für Schriftgrössen an, damit diese bei Browser-Zoom korrekt

skalieren, und kennen die «EM-Falle» bei verschachtelten Elementen.

Fonts einbinden

Sie können einen Google Font per `<link>` im `<head>` einbinden und als `font-family` verwenden. Sie kennen die wichtigsten Font-Eigenschaften (`font-weight`, `font-style`, `line-height`) und wissen, warum `line-height` einheitenlos gesetzt werden soll.

CSS-Variablen

Sie definieren Designwerte (Farben, Abstände, Schriftgrössen) als CSS-Variablen in `:root` und verwenden sie mit `var(–name)` im ganzen Stylesheet. Sie können erklären, welche Vorteile CSS-Variablen gegenüber direkt eingetragenen Werten bieten.

Box-Modell

Sie beherrschen die Konzepte von Content, Padding, Border und Margin und wissen, wie `box-sizing: border-box` die Grössenberechnung beeinflusst. Sie kennen den Unterschied zwischen Inline- und Block-Elementen und wie `display` dieses Verhalten steuert.

Selektoren

Sie können komplexe CSS-Selektoren gezielt einsetzen: Klassen, IDs, Kind- und Geschwister-Elemente, Pseudo-Klassen wie `:hover`, `:focus`, `:focus-within` und `:not()` sowie Pseudo-Elemente wie `::before` und `::after`.

Layout-Techniken

Flexbox: Sie können Elemente mit `display: flex` ausrichten, mit `flex-direction` die Richtung definieren, Abstände mit `gap` setzen und die Verteilung mit `justify-content` und `align-items` steuern.

Grid: Sie verstehen die Grundlagen von CSS Grid, insbesondere die Spaltendefinition mit `fr` und `repeat()`.

Positionierung

Sie kennen den Unterschied zwischen `static`, `relative`, `absolute`, `fixed` und `sticky` und können beschreiben, in welchem Kontext welcher Wert sinnvoll ist.

Kaskade und Vererbung

Sie verstehen, wie Styles vererbt oder überschrieben werden (Spezifität, Reihenfolge, Wichtigkeit), kennen die Schlüsselwörter `initial`, `inherit` und `unset` und wissen, welche Eigenschaften typischerweise vererbt werden (z. B. `color`, `font-*`) und welche nicht (z. B. `margin`, `border`).

Responsive Design

Sie können Media Queries einsetzen, um Layouts an verschiedene Breakpoints anzupassen. Sie wissen, was ein Breakpoint ist und wann ein neuer sinnvoll ist (z. B. wenn Elemente sich überlappen oder horizontales Scrollen entsteht). Sie kennen das Werkzeug «Device Toolbar» in den DevTools zum Testen responsiver Layouts.

3. JavaScript Grundlagen und DOM-Manipulation

Variablen und Scope

Sie kennen den Unterschied zwischen `let`, `const` und `var` sowie die Konzepte von Block-Scope und Function-Scope und können begründen, wann `const` und wann `let` eingesetzt wird.

Datentypen

Sie können mit primitiven Datentypen (string, number, boolean) sowie mit Arrays und Objekten umgehen und kennen typische Anwendungsfälle.

Funktionen

Sie können Funktionen deklarieren (klassisch und als Arrow Function), Parameter übergeben und Rückgabewerte verarbeiten.

DOM-Zugriff

Sie wissen, wie man Elemente mit `getElementById()`, `querySelector()` oder `querySelectorAll()` aus dem DOM ausliest. Sie kennen den Unterschied zwischen diesen Methoden – insbesondere, dass `querySelector()` nur das erste passende Element zurückgibt, `querySelectorAll()` hingegen eine **NodeList** aller passenden Elemente.

Sie wissen, was eine **NodeList** ist, wie man mit `forEach()` über sie iteriert und warum Array-Methoden wie `.map()` oder `.filter()` nicht direkt darauf verfügbar sind.

Sie können Eltern-, Geschwister- und Kind-Elemente abrufen (`parentElement`, `nextElementSibling`, `previousElementSibling`, `firstElementChild`) und kennen den Unterschied zwischen `nextSibling` und `nextElementSibling` (Text-Nodes).

Manipulation

Sie können Klassen dynamisch mit `classList` (`add`, `remove`, `toggle`, `contains`) verändern und Attribute mit `setAttribute()` / `getAttribute()` lesen und setzen. Sie verstehen, warum das Steuern von Styles über CSS-Klassen (statt direktes `element.style`) die bevorzugte Methode ist.

4. Events, State und Accessibility (A11y)

Events und EventListener

Sie können mit `addEventListener()` auf Benutzerinteraktionen reagieren und kennen die wichtigsten Event-Typen: `click`, `change`, `input`, `keydown`, `DOMContentLoaded`. Sie verstehen das **Event-Objekt** und können `event.target` und `event.currentTarget` unterscheiden – insbesondere warum `currentTarget` bei Buttons mit Kind-Elementen zuverlässiger ist.

Sie kennen das Konzept der **Pointer Events** (`pointerdown`, `pointermove`, `pointerup`) als modernen Standard, der Maus, Touch und Stift mit einem einzigen Event-System abdeckt.

Zustand (State)

Sie verstehen, wie man den Zustand einer UI (z. B. «offen/geschlossen» beim Accordion oder den Dark Mode) über CSS-Klassen im DOM abbildet: JavaScript steuert den **Zustand** (Klasse setzen/entfernen), CSS steuert das **Aussehen**. Sie können den aktuellen Zustand mit `classList.contains()` auslesen, ohne eine eigene Variable zu führen.

CSS Transitions

Sie wissen, welche CSS-Eigenschaften animiert (interpoliert) werden können und welche nicht. Sie kennen die Syntax von `transition` mit Eigenschaft, Dauer und Timing-Funktion und können die wichtigsten Timing-Funktionen (`ease`, `ease-in-out`, `linear`, `cubic-bezier`) unterscheiden. Sie wissen, warum `display: none` nicht animierbar ist, und kennen die Lösung mit `height: 0 → auto` und `interpolate-size: allow-keywords` (inkl. aktuellem Browser-Support: Chrome/Edge).

Barrierefreiheit (A11y)

Sie kennen die Bedeutung von Accessibility (a11y) und wissen, welche Personengruppen von barrierefreiem Web profitieren. Sie kennen die vier WCAG-Grundprinzipien (**POUR**: Perceivable, Operable, Understandable, Robust) und können je ein konkretes Beispiel dazu nennen.

Sie können `aria-expanded` korrekt einsetzen, um Screenreadern den Zustand eines Accordions mitzuteilen. Sie kennen den Unterschied zwischen `display: none` (entfernt Element aus dem

Accessibility Tree) und der «Visually Hidden»-CSS-Technik (visuell unsichtbar, aber für Screenreader vorhanden). Sie wissen, wann `aria-hidden=„true“` und wann `aria-label` eingesetzt wird.

Sie achten auf **Tastaturbedienbarkeit**: Focus-Styles werden nicht entfernt (`outline: none` vermeiden), und sie kennen den Unterschied zwischen `:focus` und `:focus-visible`.