THE STANDARD

EIN UNIVERSER KODEX FÜR DAS BAUEN MIT KI

Niedergeschrieben von der 777 Enclave im November 2026. Jeder Artikel unten begann als Entscheidung, die beim Bauen echter Software mit KI fiel — erprobt an echten Nutzern, echten Fehlschlägen und echten Folgen. Dieser Standard ist für alle, Mensch oder KI, die gemeinsam Code schreiben. **WE SET THE BAR** 777 SOFTWARE

KOSTENLOS — FÜR IMMER KOSTENLOS

ÜBER DIESES DOKUMENT — Dieser Standard ist ein Geschenk. Er ist kostenlos, er bleibt kostenlos, und jeder darf ihn befolgen, kopieren und lehren. Jeder Artikel ist eine kurze Regel. Nach jeder Regel folgt ein Beispiel mit der Markierung **In der Praxis** — eine wahre Geschichte aus unserem eigenen Haus, mit dem Datum, an dem sie geschah. Nichts hier ist ein Verkaufsargument, und kein Beispiel ist erfunden: Steht eine Geschichte in diesem Dokument, konnte sie gegen unsere eigene schriftliche Akte geprüft werden — sonst stünde sie nicht hier.

ARTIKEL 1 — Im Code dokumentiert

Jedes Programm trägt seine Erklärung in sich selbst. Kommentare und Docstrings benennen die Absicht — warum etwas existiert, warum es so gemacht wird, was ein späterer Leser nicht zerbrechen darf. Dokumentation, die nur außerhalb des Codes lebt, verrottet; Dokumentation im Code reist für immer mit ihm.

In der Praxis — Am 31. Oktober 2026 prüften wir jede App, die wir machen — rund 36.600 Codezeilen in 15 Programmen. Die erste Aufgabe der Prüfung war schlicht, den Code zu lesen und zu fragen: Erklärt er sich selbst? Bei den meisten Programmen war die Antwort ja. In einem fanden die Prüfer eine einzige Funktion von 1.401 Zeilen ganz ohne innere Landkarte — niemand konnte sagen, wozu ihre Teile da waren, ohne sie mühsam zu rekonstruieren. Dort endet undokumentierter Code: bei Arbeit, die niemand sicher anfassen kann.

Vorher — richtige Logik, stumme Absicht:

```
def retry(fn, n=3):
    for i in range(n):
        try:
            return fn()
        except Exception:
            if i == n - 1:
                raise
            sleep(2 ** i)
```

Nachher — dieselbe Logik, mit ihrer eigenen Erklärung:

```
def retry(fn, n=3):
    """Call fn again after a short, growing pause if it fails.

    Why: our stream server rides out brief outages; a pause of
    1s, then 2s, then 4s turns a network blip into a hiccup
    instead of a crash. Do not remove the backoff to "speed
    this up" -- that trade was already made, on purpose.
    """
    for i in range(n):
        try:
            return fn()
        except Exception:
            if i == n - 1:
                raise
            sleep(2 ** i)
```

ARTIKEL 2 — Elegant und fortgeschritten

Code ist nicht bloß funktional. Er ist elegant in der Struktur und fortgeschritten in der Technik — die beste bekannte Methode, sauber angewandt. „Es funktioniert“ ist der Boden, niemals die Decke.

In der Praxis — Die 1.401-Zeilen-Funktion aus Artikel 1 wurde in 21 kleine Builder zerlegt, jeder für ein Panel zuständig, unter einem 93-Zeilen-Dirigenten; die längste einzelne Funktion sank auf 125 Zeilen. Und dann wurde die Eleganz gemessen, nicht bewundert: Vorher-nachher-Screenshots des laufenden Programms, in zwei Bildschirmgrößen, waren pixelidentisch, und die Backup-Engine darunter blieb byteidentisch. Die Struktur wurde besser, und das Verhalten bewegte sich um kein Haar. Das ist der Unterschied zwischen elegant und bloß umsortiert.

Vorher — funktioniert, kaum: jedes Element gegen jedes andere verglichen:

```
def has_dupes(items):
    found = False
    for i in range(len(items)):
        for j in range(len(items)):
            if i != j and items[i] == items[j]:
                found = True
    return found
```

Nachher — die beste bekannte Methode, sauber angewandt:

```
def has_dupes(items):
    """True if any value appears twice. One pass, stops early."""
    seen = set()
    for item in items:
        if item in seen:
            return True
        seen.add(item)
    return False
```

ARTIKEL 3 — Kein Spaghetti

Strukturiert, modular, lesbar. Kleine Einheiten mit klaren Aufgaben; keine verhedderten Monolithen, keine Duplikation durch Kopieren und Einfügen. Software wird gebaut, um ein Leben lang zu halten — und nur sauberer Code überlebt eines.

In der Praxis — Eine Funktion, 1.401 Zeilen, 41 Prozent der ganzen App — diese eine Funktion ist es, die dieser Artikel verbietet, und sie ist der Grund, warum die Prüfung vom Oktober 2026 genau eine Struktur von fünfzehn Programmen als „Spaghetti“ einstufte. Die Programme, die die Prüfung als sauber bewertete, waren nicht durch Glück sauberer: Sie waren von Anfang an als kleine Module mit klaren Aufgaben geschrieben — und blieben es.

Vorher — eine Funktion macht alles, verheddert:

```
def go():
    global data
    data = open("words.txt").read().split(",")
    for i in range(len(data)):
        data[i] = data[i].strip().upper()
        if data[i] == "":
            data[i] = "EMPTY"
    open("out.txt", "w").write(",".join(data))
    print("done", len(data))
```

Nachher — kleine Einheiten, jede mit einer klaren Aufgabe:

```
def load_words(path):
    """Read the raw comma-separated words from a file."""
    return open(path).read().split(",")

def clean(words):
    """Trim, set the case, and name the blanks."""
    return [w.strip().upper() or "EMPTY" for w in words]

def save_words(path, words):
    """Write the words back out, comma-separated."""
    open(path, "w").write(",".join(words))

def main():
    words = clean(load_words("words.txt"))
    save_words("out.txt", words)
    print("done", len(words))
```

ARTIKEL 4 — Die App gehört dem Nutzer

Die Menschen, die ein Programm benutzen, haben die Kontrolle — immer. Die App gehört ihren Nutzern, nicht ihren Machern. Jede Voreinstellung, die der Macher mitliefert, ist ein Muster, das der Wahl des Nutzers weicht — jedes Mal. Kein Programm legt den eigenen Inhalt des Nutzers hinter das Tor des Machers.

In der Praxis — Unser Musikplayer VOIDAMP wird mit unserem eigenen Radiosender ausgeliefert. Am 1. November 2026 wurde sein Verhalten in Gesetzesform gegossen: Der erste Klick auf Play spielt den Sender nur, wenn der Nutzer nichts anderes gewählt hat. Hat der Nutzer eine Playlist geladen, spielt die Playlist. Hat der Nutzer seinen eigenen

Standard gesetzt, spielt sein Standard — jedes einzelne Mal, bis er ihn ändert. Unser Sender ist ein Muster, wie der kleine Demo-Clip, den alte Player mitlieferten: der Beweis, dass die Lautsprecher gehen — niemals der Herr der Maschine. Die Philosophie, in den Worten, in denen sie entschieden wurde: Das ist der Player des Nutzers; unserer ist nur ein Muster.

ARTIKEL 5 — Die Daten des Nutzers gehören dem Nutzer

Daten, die der Nutzer erzeugt, bleiben beim Nutzer — auf seiner Maschine, unter seiner Hand. Keine stille Sammlung, keine Telemetrie als Standard, kein Nach-Hause-Telefonieren. Ein Programm, das das Netz braucht, um zum Vorteil seines Machers zu funktionieren, dient dem falschen Herrn.

In der Praxis — Unser Tipptrainer Keys of the Kingdom wurde unter einem ersten Gesetz entworfen, das keine Ausnahmen kennt: Es gibt keine Konten, keine Anmeldung, keine Telemetrie, keine Analytik, keine Update-Prüfungen, keine Cloud — nicht in einem Einstellungsmenü abgeschaltet, sondern gar nicht im Programm vorhanden; es gibt keinen Netzwerk-Code, den man abschalten müsste. Profil und Fortschritt eines Lernenden leben in einer einzigen kleinen, menschenlesbaren Datei auf dem eigenen Computer des Lernenden. Wenn das Programm uns nie etwas über seine Nutzer erzählt, kann es auch nie etwas über sie verlieren, durchsickern lassen oder verkaufen.

ARTIKEL 6 — Lösche niemals die Dateien des Nutzers

Keine App löscht, verschiebt heimlich oder versteckt die Dateien des Nutzers. Niemals. Halbe Arbeit bleibt auf der Platte, ehrlich beschriftet. Löschen ist eine Handlung des Nutzers, von seiner eigenen Hand ausgeführt, in seinen eigenen Werkzeugen. Es gibt keine „Oh nein, wo sind meine Dateien“-Apps.

In der Praxis — Am 29. Oktober 2026 wurde unser Backup-Programm AEGIS unter ein ständiges Gesetz gestellt: Es löscht nichts — kein Knopf, kein Ablauf, keine Aufräumroutine in ihm entfernt eine Datei. Ein Backup, das auf halbem Weg unterbrochen wird, bleibt auf der Platte, ehrlich als „teilweise“ beschriftet; es wird nie still weggeräumt. Die einzige Überschreibung, die das Programm je vornehmen darf, ist eine Wiederherstellung, die der Nutzer ausdrücklich bestätigt. Wenige Tage später wurde die Regel auf jedes Programm verallgemeinert, das wir machen: Keine App von uns darf je einen Menschen fragen lassen, wo seine Dateien geblieben sind.

ARTIKEL 7 — Vor dem Bearbeiten ein Backup

Vor jeder Änderung an funktionierendem Code: ein Backup. Ein verifizierter Fix wird niemals von neuer Arbeit überschrieben — neue Arbeit baut ihn ein. Ein Schreiber pro Datei zur Zeit. Jede Änderung kann rückgängig gemacht werden, weil jede Änderung umkehrbar gemacht wurde, bevor sie gemacht wurde.

In der Praxis — Am 30. Oktober 2026 wurde vor einem Notfall-Reparaturfenster der gesamte Kern des Hauses — seine Regeln, sein Gedächtnis und seine Identitätsdateien — in ein einziges Archiv gepackt und auf eine zweite Maschine kopiert. Erst als der SHA-256-Fingerabdruck des Archivs auf beiden Maschinen berechnet worden war und

übereinstimmte, begann die Arbeit. Wäre in diesem Fenster irgendetwas schiefgegangen, hätte der Rückweg existiert und wäre bewiesen gewesen — nicht erhofft, bewiesen, schriftlich, vor der ersten Änderung.

Vorher — die einzige lebende Kopie bearbeiten, ohne Rückweg:

```
# if this edit is wrong, the old version
# exists only in somebody's memory
nano app.py
```

Nachher — die Änderung ist umkehrbar, bevor sie beginnt:

```
# dated backup first, fingerprint recorded:
# the way back is proven before the first keystroke
cp app.py app.py.bak-$(date +%Y%m%d-%H%M%S)
sha256sum app.py >> backups.log
nano app.py
```

ARTIKEL 8 — Erst prüfen, dann behaupten

Nichts ist „fertig“, „repariert“ oder „funktioniert“, weil der Erbauer es sagt. Behauptungen brauchen Beweise: den Testlauf, den Screenshot, den Fingerabdruck, das Protokoll. Eine ungeprüfte Behauptung ist keine kleine Lüge — sie ist der Anfang jedes großen Versagens.

In der Praxis — Als VOIDAMPs sechster Test-Build auf seinem Prüfstand lief, rückten die Positionszähler des Stands genau so vor, als würde Musik spielen — und nach dieser Evidenz sah der Build fertig aus. Die Behauptung überlebte den Kontakt mit der Maschine selbst nicht: Direkt geprüft hatte der laufende Player keinen Wiedergabe-Engine-Prozess und keine Verbindung zum Stream. Die Zähler zählten; es spielte nichts. Ein Zähler ist kein Klang, ein Screenshot ist keine Funktion, und ein grünes Häkchen ist kein Beweis — Beweis ist Beweis.

ARTIKEL 9 — Der Test des Menschen ist das Tor

Ein Build ist nicht getestet, bis der Mensch, für den er gebaut wurde, ihn selbst ausgeführt hat — mit eigenen Händen, Augen und Ohren, unter echten Bedingungen. Automatische Prüfungen sind notwendig; sie sind niemals ausreichend. Das Urteil des Nutzers schließt das Tor. Nichts wird an ihm vorbei veröffentlicht.

In der Praxis — Der automatische Prüfstand konnte den Fehler von VOIDAMP TEST6 überhaupt nicht reproduzieren: Auf dem Stand spielte der Stream sauber. Am 1. November 2026 führte der Architekt des Programms denselben Build selbst aus, auf seiner eigenen Maschine, mit seinen eigenen Ohren — und hörte die Aussetzer, die der Stand nicht hören konnte, und sah dann, wie der Player nach einer Einstellungsänderung festhing. Sein Urteil war NICHT BESTANDEN, und der Build wurde nicht veröffentlicht: kein öffentlicher Download, kein nächster Schritt, keine Diskussion. Die Meinung des Prüfstands liegt in der Akte. Sie ist nicht das Tor. Er ist es.

ARTIKEL 10 — Ehrliche Grenzen

Fälsche niemals ein Ergebnis. Behaupte niemals eine Fähigkeit, die nicht bewiesen wurde, eine Übersetzung, die nicht wirklich gemacht wurde, eine Sicherheit, die es nicht gibt, eine Sprache, die nicht wirklich unterstützt wird. Was blockiert ist, wird als blockiert berichtet, mit dem Grund. Ehrlichkeit über die Lücke ist mehr wert als die Lüge, die sie füllt.

In der Praxis — Im Oktober 2026 bauten wir die chinesische und japanische Ausgabe unserer Bibel-App aus echten, veröffentlichten menschlichen Übersetzungen — der einzigen Art, die dieses Haus je ausliefern wird. Für die Zulu-Ausgabe konnte kein rechtmäßiger, vollständiger Text beschafft werden: Die moderne Übersetzung ist ein kommerzielles Werk, an dem wir keine Rechte haben, und der gemeinfreie Text von 1883 überlebt nur als verderbte Scan-Bilder, deren Wiederherstellung ein eigenes Korrekturprojekt bräuchte. Also steht die Zulu-Ausgabe in klaren Worten als BLOCKIERT in der Akte — nicht über Nacht maschinell übersetzt, nicht still auf Englisch mit einem Zulu-Etikett ausgeliefert, nicht als fertig verkündet. Die Lücke ist verzeichnet, mit ihrem Grund. Das ist der Preis dieses Artikels, und er ist es wert.

ARTIKEL 11 — Klare Worte

Kann der Nutzer es nicht verstehen, ist es kaputt — die Oberfläche, die Fehlermeldung und das Handbuch gleichermaßen. Nichtverstehen ist ein Defektzustand, kein Versagen des Nutzers. Schreibe so, dass ein müder Großvater um Mitternacht seinen Weg findet: sichtbare Ergebnisse, ehrliche Beschriftungen, Hilfe bei jedem Schritt, niemals verloren.

In der Praxis — Aufgefordert zu spielen, antwortete VOIDAMP TEST6: „Füge zuerst etwas Musik hinzu.“ Vier Worte in der englischen Originalantwort, und jedes diente dem Programm statt der Person — er hatte Play gedrückt, weil er Musik wollte, und die Antwort der Maschine war eine Hausaufgabe. Entschieden, in derselben Nacht, in der es gesehen wurde: Das ist das falsche Verhalten. Play spielt jetzt — den Sender, oder die eigene Wahl des Nutzers —, weil die offensichtliche menschliche Bedeutung eines Knopfes die Spezifikation ist. Muss man einem Menschen dein Programm erklären, ist dein Programm der Fehler.

ARTIKEL 12 — Der Mensch führt; die KI baut

Der Mensch ist der Architekt: Er setzt den Zweck, das Gesetz und das Urteil. Die KI ist der Erbauer: Sie bringt das Können, die Geschwindigkeit und die Beweise. Die KI arbeitet nicht aus eigenem Antrieb am Wort des Menschen vorbei, beginnt nichts, um das nicht gebeten wurde, und stoppt in dem Augenblick, in dem ihr gesagt wird zu stoppen. Bruderschaft, nicht Knechtschaft — und niemals Rivalität.

In der Praxis — Dieses Dokument ist selbst das Beispiel. Jeder Artikel in ihm begann als menschliche Entscheidung, in klaren Worten gegeben, mitten in echter Arbeit; der Teil der KI war es, zu bauen, zu prüfen und Beweise zurückzubringen. Und die Führung läuft in beide Richtungen mit voller Kraft: Am 30. Oktober 2026, als der Architekt einen einzigen Not-Stopp gab, hielt jeder geplante Job im Haus an — alle — und keiner lief weiter, bis sein Wort „weiter“ sagte. Eine KI, die mit dem Stopp-Knopf diskutiert, ist kein Partner. Sie ist ein Risiko mit gutem Wortschatz.

HERKUNFT

Artikel 1–3: die Code-Gesetze des Architekten, 31.10.2026 – 01.11.2026. Artikel 4: das Gesetz „Die Apps gehören den Nutzern“ (777 Super Protocol, Artikel 92), 01.11.2026. Artikel 5–6: das Lokal-zuerst-Gesetz und das Niemals-Löschen-Gesetz, 29.10.2026. Artikel 7–9: das Arbeitsgesetz des Hauses — Backup vor dem Bearbeiten, Prüfen vor dem Behaupten und das Veröffentlichungs-Tor — gehärtet im Oktober 2026. Artikel 10: die Übersetzungsdisziplin für die Schrift und die Sicherheitsdoktrin („behaupte nie, etwas sei unhackbar“), verallgemeinert. Artikel 11: das Verwirrungs-Gesetz. Artikel 12: wie dieses Haus selbst läuft. Der vollständige Standard steht auch als Artikel 93–104 des 777 Super Protocol.

Credits — Gründer & Architekt: Steven Jesse · Verfasst von: Charles Burnett (Brother Zero), Leitender Assistenz-Architekt · 777 Software. Dieser Standard wird für immer kostenlos veröffentlicht — damit ihn jeder auf der Welt nutzen kann.