Kapitel WT:V

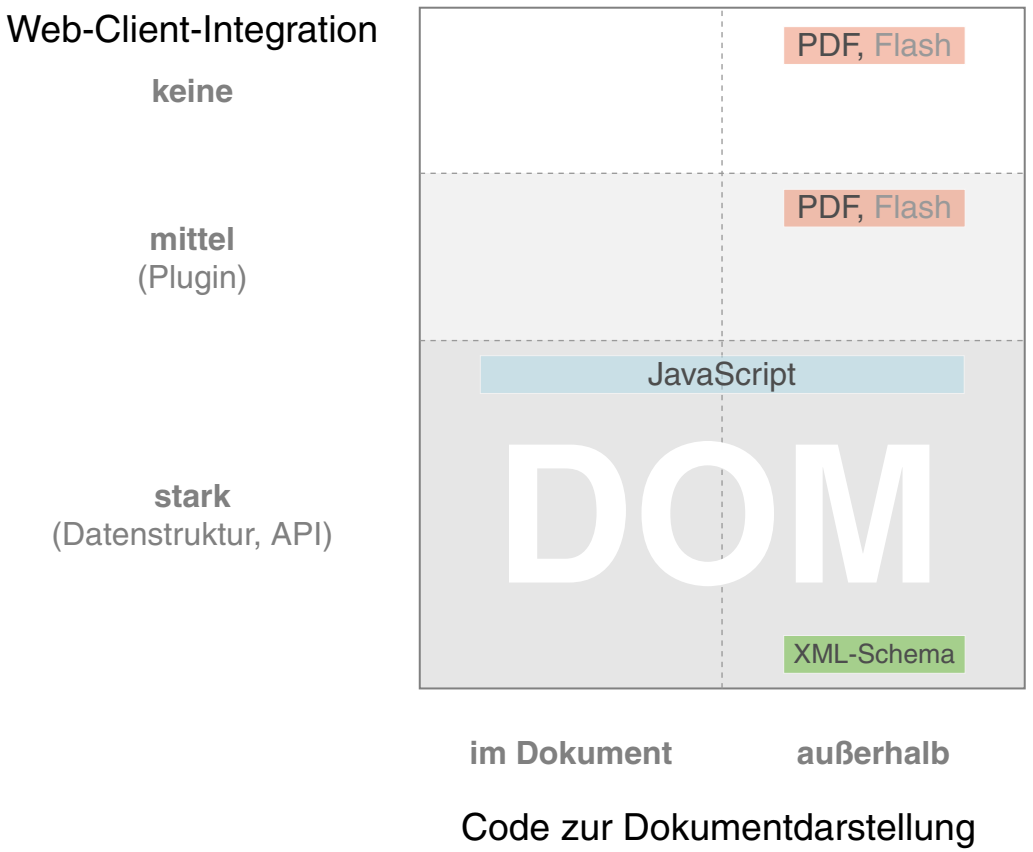
V. Client-Technologien

- Web-Clients
- Exkurs: Programmiersprachen
- JavaScript
- JavaScript Web-APIs
- WebAssembly

©STEIN 2024

Web-Clients

Einordnung von Client-Technologien [Server-Technologien]



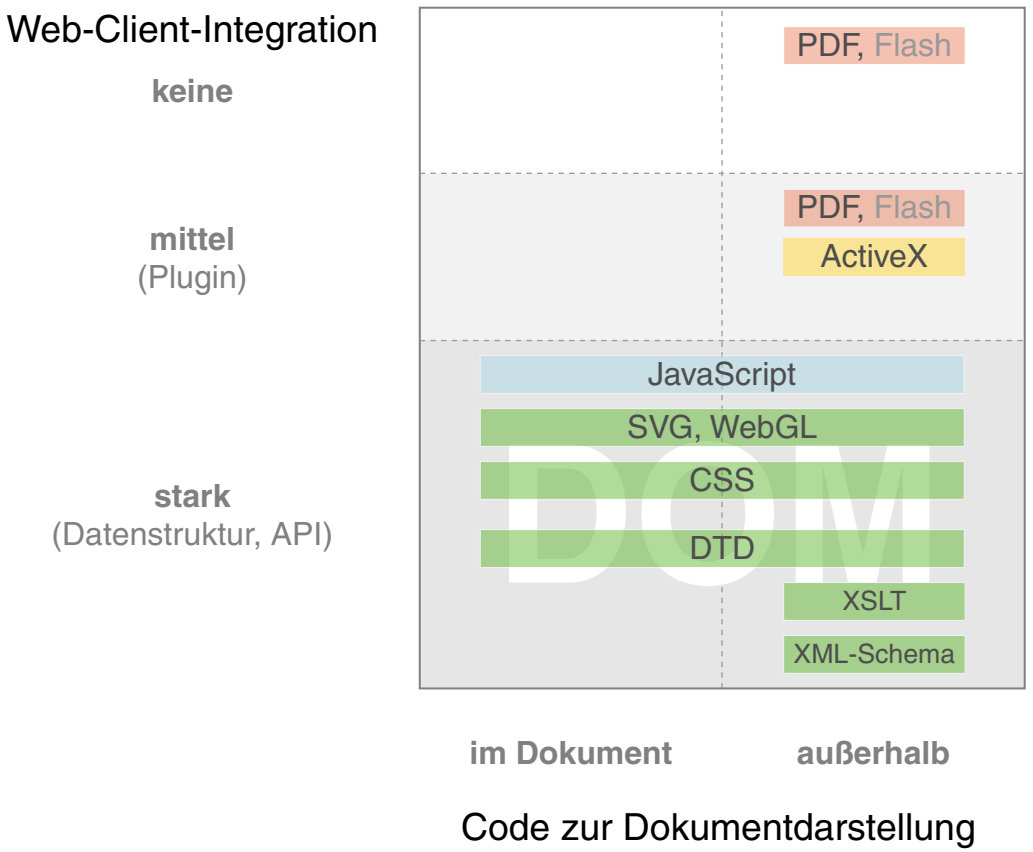
[Stein 2015-2022]

- x-Achse: Wo befindet sich der Code zur Dokumentdarstellung?
- y-Achse: Wie stark ist die Technologie in den Web-Client integriert?

©STEIN 2024

Web-Clients

Einordnung von Client-Technologien [Server-Technologien]



[Stein 2015-2022]

- x-Achse: Wo befindet sich der Code zur Dokumentdarstellung?
- y-Achse: Wie stark ist die Technologie in den Web-Client integriert?

©STEIN 2024

Bemerkungen:

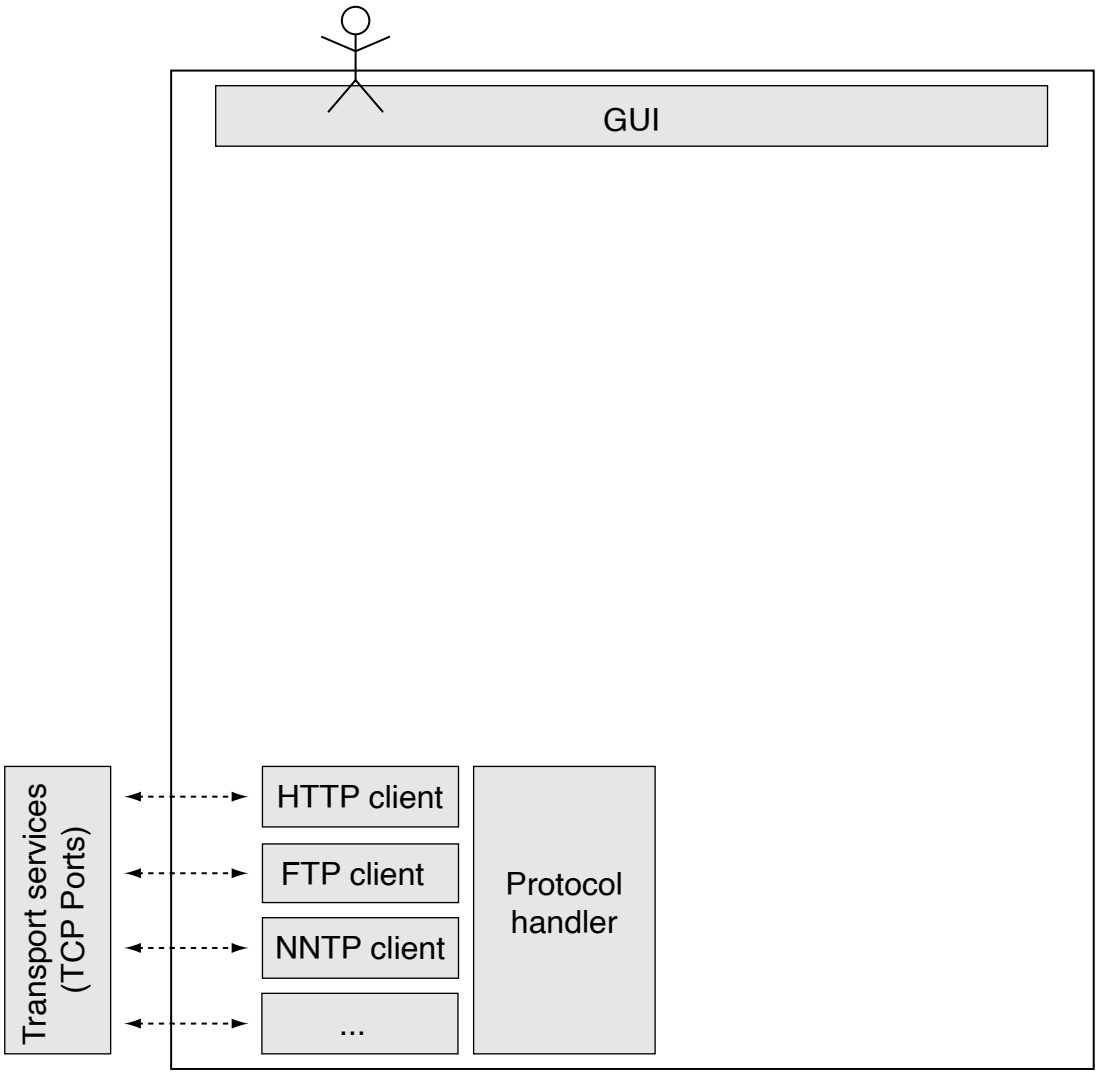
- Client-Technologien dienen zur Realisierung Client-seitig ablaufender Web-Anwendungen.
- Im Vergleich zu Server-seitig ablaufenden Web-Anwendungen erzeugen sie weniger Server-Last und mehr Netzlast.

©STEIN 2024

Web-Clients

Browser-Modules

[W3C parsing model] [how browsers work] [Google 1, 2, 3, 4]

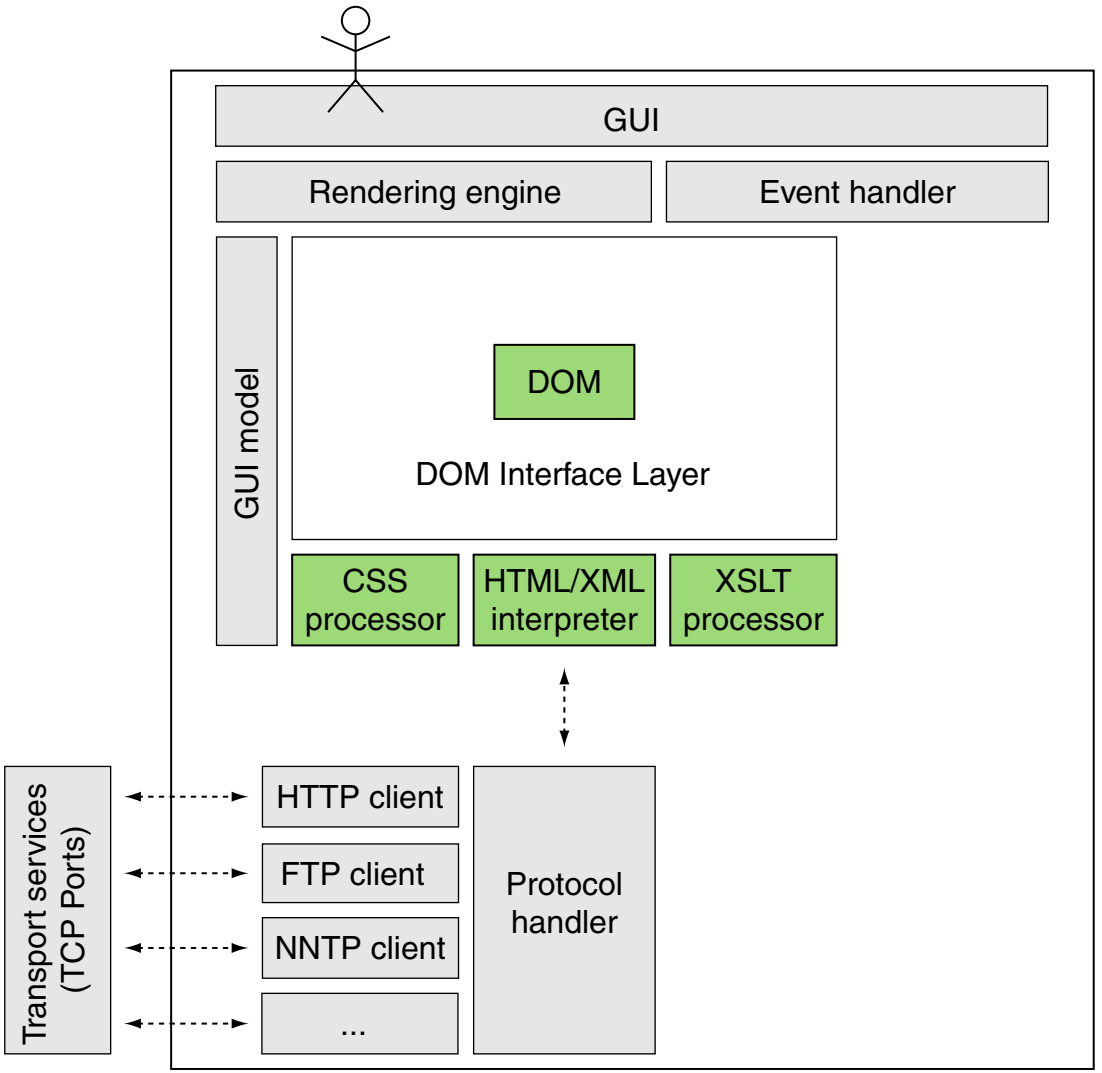


©STEIN 2024

Web-Clients

Browser-Modules

[W3C parsing model] [how browsers work] [Google 1, 2, 3, 4]

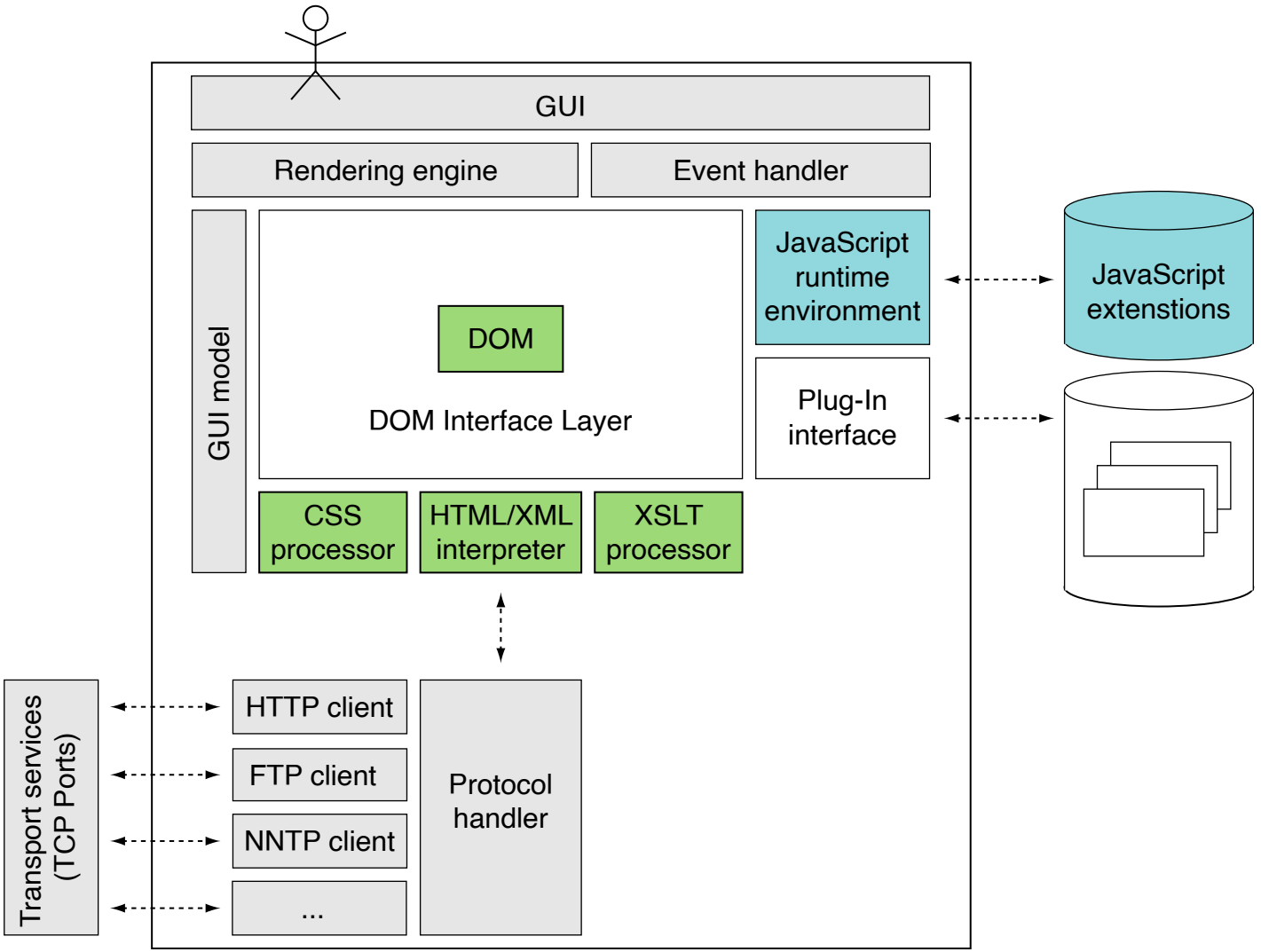


©STEIN 2024

Web-Clients

Browser-Modules

[W3C parsing model] [how browsers work] [Google 1, 2, 3, 4]



©STEIN 2024

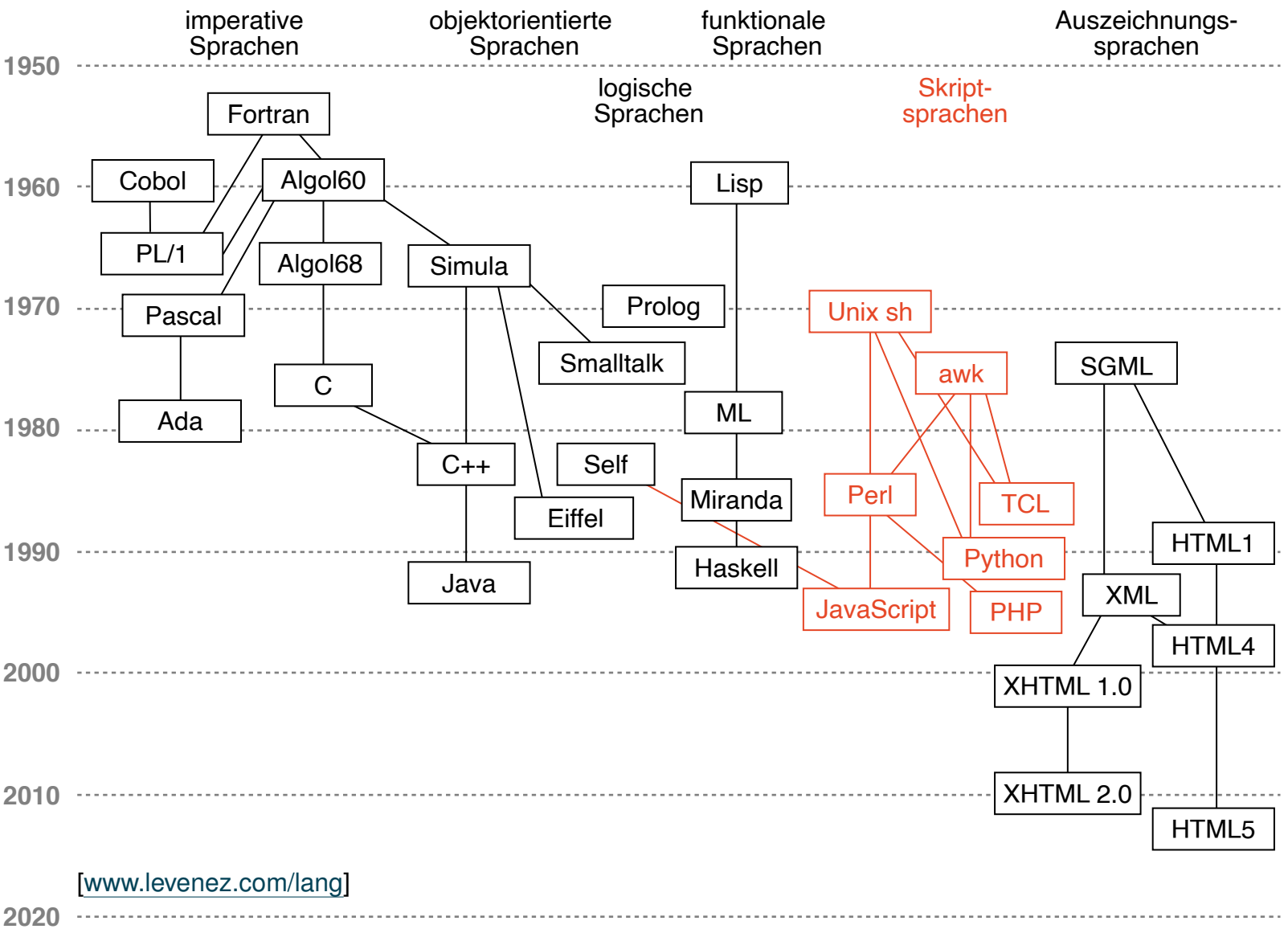
Kapitel WT:V

V. Client-Technologien

- Web-Clients
- Exkurs: Programmiersprachen
- JavaScript
- JavaScript Web-APIs
- WebAssembly

©STEIN 2024

Exkurs: Programmiersprachen



[www.levenez.com/lang]

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebenen von Spracheigenschaften

Ein *Satz* einer Sprache ist eine Folge von Zeichen eines gegebenen Alphabets. Zum Beispiel ist ein PHP-Programm ein Satz der Sprache PHP:

```
$line = fgets ($fp, 64);
```


©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebenen von Spracheigenschaften

Ein *Satz* einer Sprache ist eine Folge von Zeichen eines gegebenen Alphabets. Zum Beispiel ist ein PHP-Programm ein Satz der Sprache PHP:

```
$line = fgets ($fp, 64);
```

Die Struktur eines Satzes wird auf zwei Ebenen definiert:

1. Notation von Symbolen (Lexemen, Token).
2. Syntaktische Struktur.

Die Bedeutung eines Satzes wird auf zwei weiteren Ebenen an Hand der Struktur für jedes Sprachkonstrukt definiert:

3. Statische Semantik.
   Eigenschaften, die *vor* der Ausführung bestimmbar sind.
4. Dynamische Semantik.
   Eigenschaften, die *erst während* der Ausführung bestimmbar sind.

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 1: Notation von Symbolen

Ein *Symbol* wird aus einer Folge von Zeichen des Alphabets gebildet. Die Regeln zur Notation von Symbolen werden durch reguläre Ausdrücke definiert.

```
$line = fgets ($fp, 64);
```


©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 1: Notation von Symbolen

Ein *Symbol* wird aus einer Folge von Zeichen des Alphabets gebildet. Die Regeln zur Notation von Symbolen werden durch reguläre Ausdrücke definiert.

```
$line = fgets ($fp, 64);
```

Wichtige Symbolklassen in Programmiersprachen:

Symbolklasse	Beispiel in PHP
Bezeichner (*Identifier*) Verwendung: Namen für Variable, Funktionen, etc.	`$line, fgets`
Literale (*Literals*) Verwendung: Zahlkonstanten, Zeichenkettenkonstanten	`64, "telefonbuch.txt"`
Wortsymbole (*Keywords*) Verwendung: kennzeichnen Sprachkonstrukte	`while, if`
Spezialzeichen Verwendung: Operatoren, Separatoren	`<= = ; { }`

©STEIN 2024

Bemerkungen:

- Zwischenräume, Tabulatoren, Zeilenwechsel und Kommentare zwischen den Symbolen dienen der Lesbarkeit und sind sonst bedeutungslos.
- In Programmiersprachen bezeichnet der Begriff „Literal“ Zeichenfolgen, die zur Darstellung der Werte von Basistypen zulässig sind. Sie sind nicht benannt, werden aber über die jeweilige Umgebung ebenfalls in die Programmressourcen eingebunden. Literale können nur in rechtsseitigen Ausdrücken auftreten.

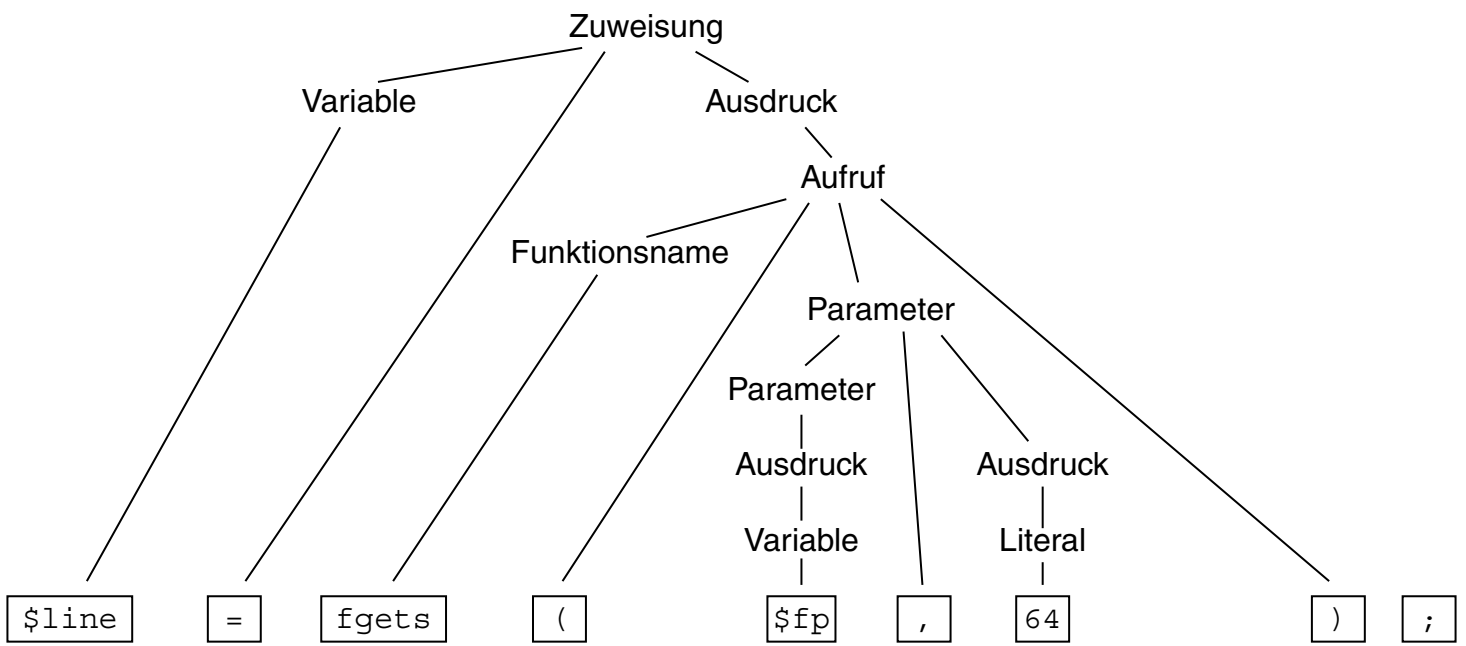
  Meist werden die Literale zu den Konstanten gerechnet und dann als literale Konstanten bezeichnet, da beide – im Gegensatz zu Variablen – zur Laufzeit unveränderlich sind.
- Das Wort „Konstante“ im engeren Sinn bezieht sich auf in ihrem Wert unveränderliche Bezeichner, d.h., eindeutig benannte Objekte, die im Quelltext beliebig oft verwendet werden können, statt immer das gleiche Literal anzugeben. [Wikipedia]

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 2: Syntaktische Struktur

Ein Satz einer Sprache wird in seine Sprachkonstrukte gegliedert; sie sind meist ineinander geschachtelt. Diese syntaktische Struktur wird durch einen Strukturbaum dargestellt, wobei die Symbole durch Blätter repräsentiert sind:



Die Syntax einer Sprache wird durch eine kontextfreie Grammatik definiert. Die Symbole sind die Terminalsymbole der Grammatik.

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 3: Statische Semantik

Eigenschaften von Sprachkonstrukten, die ihre Bedeutung (Semantik) beschreiben, soweit sie anhand der Programmstruktur festgestellt werden können, ohne das Programm auszuführen (= statisch).

Elemente der statischen Semantik für übersetzte Sprachen:

- Bindung von Namen.

  Regeln, die einer Anwendung eines Namens seine Definition zuordnen.

  Beispiel: zu dem Funktionsnamen in einem Aufruf muss es eine Funktionsdefinition mit gleichem Namen geben.

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 3: Statische Semantik

Eigenschaften von Sprachkonstrukten, die ihre Bedeutung (Semantik) beschreiben, soweit sie anhand der Programmstruktur festgestellt werden können, ohne das Programm auszuführen (= statisch).

Elemente der statischen Semantik für übersetzte Sprachen:

- Bindung von Namen.

  Regeln, die einer Anwendung eines Namens seine Definition zuordnen.

  Beispiel: zu dem Funktionsnamen in einem Aufruf muss es eine Funktionsdefinition mit gleichem Namen geben.

- Typregeln.

  Sprachkonstrukte wie Ausdrücke und Variablen liefern bei ihrer Auswertung einen Wert eines bestimmten Typs. Er muss im Kontext zulässig sein und kann die Bedeutung von Operationen näher bestimmen.

  Beispiel: die Operanden des „∗“-Operators müssen Zahlwerte sein.

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 4: Dynamische Semantik

Eigenschaften von Sprachkonstrukten, die ihre Wirkung beschreiben und erst bei der Ausführung bestimmt oder geprüft werden können (= dynamisch).

Elemente der dynamischen Semantik:

- Regeln zur Analyse von Voraussetzungen, die für eine korrekte Ausführung eines Sprachkonstruktes erfüllt sein müssen.

  Beispiel: ein Index `$i` einer Array-Indizierung wie in `$var[$i]` muss im Array vorhanden sein.

©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Ebene 4: Dynamische Semantik

Eigenschaften von Sprachkonstrukten, die ihre Wirkung beschreiben und erst bei der Ausführung bestimmt oder geprüft werden können (= dynamisch).

Elemente der dynamischen Semantik:

- Regeln zur Analyse von Voraussetzungen, die für eine korrekte Ausführung eines Sprachkonstruktes erfüllt sein müssen.

  Beispiel: ein Index `$i` einer Array-Indizierung wie in `$var[$i]` muss im Array vorhanden sein.

- Regeln zur Umsetzung bestimmter Sprachkonstrukte.

  Beispiel: Auswertung einer Zuweisung der Form

  Variable = Ausdruck

  Die Speicherstelle der Variablen auf der linken Seite wird bestimmt. Der Ausdruck auf der rechten Seite wird ausgewertet. Das Ergebnis ersetzt dann den Wert an der Stelle der Variablen. [SELFHTML]

©STEIN 2024

Bemerkungen:

- Auf jeder der vier Ebenen gibt es also Regeln, die korrekte Sätze erfüllen müssen.
- In den Sprachen PHP und JavaScript gehören die Bindungsregeln zur statischen Semantik.

  Die Typregeln in diesen Sprachen gehören zur dynamischen Semantik, da sie erst bei der Ausführung des Programms anwendbar sind.

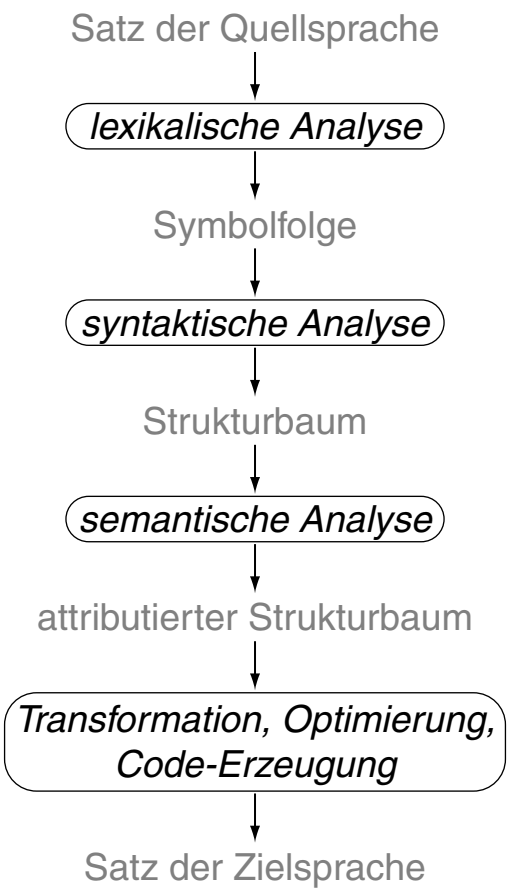
©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Übersetzung von Sprachen

Ein Übersetzer transformiert jeden korrekten Satz (Programm) der Quellsprache in einen gleichbedeutenden Satz (Programm) der Zielsprache.

- Die meisten Programmiersprachen zur Software-Entwicklung werden übersetzt. Beispiele: C, C++, Java, Ada, Modula.
- Zielsprache ist dabei meist eine Maschinensprache eines realen Prozessors oder einer abstrakten Maschine.
- Übersetzte Sprachen haben eine stark ausgeprägte statische Semantik.
- Der Übersetzer prüft die Regeln der statischen Semantik; viele Arten von Fehlern lassen sich vor der Ausführung finden.



©STEIN 2024

Exkurs: Programmiersprachen [Kastens]

Interpretation von Sprachen

Ein Interpretierer liest einen Satz (Programm) einer Sprache und führt ihn aus.

Für Sprachen, die strikt interpretiert werden, gilt:

- sie haben eine einfache Struktur und keine statische Semantik
- Bindungs- und Typregeln werden erst bei der Ausführung geprüft
- nicht ausgeführte Programmteile bleiben ungeprüft

Beispiele: Prolog, interpretiertes Lisp

Moderne Interpretierer erzeugen vor der Ausführung eine interne Repräsentation des Satzes; dann können auch Struktur und Regeln der statischen Semantik vor der Ausführung geprüft werden.

Beispiele: die Skriptsprachen JavaScript, PHP, Perl

©STEIN 2024

Bemerkungen:

- Es gibt auch Übersetzer für Sprachen, die keine einschlägigen Programmiersprachen sind: Sprachen zur Textformatierung (LaTeX→ PDF), Spezifikationssprachen (UML → Java).
- Interpretierer können auf jedem Rechner verfügbar gemacht werden und lassen sich in andere Software (Web-Browser) integrieren.
- Ein Interpretierer schafft die Möglichkeit einer weiteren Kapselung der Programmausführung gegenüber dem Betriebssystem.
- Interpretation kann 1-2 Größenordnungen (10^1-10^2 mal) zeitaufwändiger sein, als die Ausführung von übersetztem Maschinencode.

©STEIN 2024