GeoCSS Schulung

- Einführung GeoCSS
- Erstellung von Vektorsymbolen (Punkt/Linien/Flächen)
- Klassifikation von Symbolen
- Operatoren und Bedingungen in der Symbolerstellung
- Beschriftung von Symbolen
- Erstellung von Legenden
- Raster-/Gridsymbole
- Alternative Geometrien

Onlinehilfe + Schulungsunterlagen

- https://www.cardogis.com/iwan7css
- http://www.cardogis.com/infomaterialien

GeoCSS

- An CSS (Cascading Style Sheets) angelehnte Syntax
- Durch IDU weiterentwickelt um Geofunktionen
- IWAN7 Ebenen benötigen CSS als Symboldefinition
- Umfangreichere Gestaltungsmöglichkeiten als mit Symboleditor für Iwan6 Ebenen (Klassifikation nach zwei Spalten)
- Vermischungen von Zeichenoperationen pro Geometrietyp (Bspw. Polygon -> bestand aus Fläche und Linie) nicht mehr gegeben

Einführung CSS

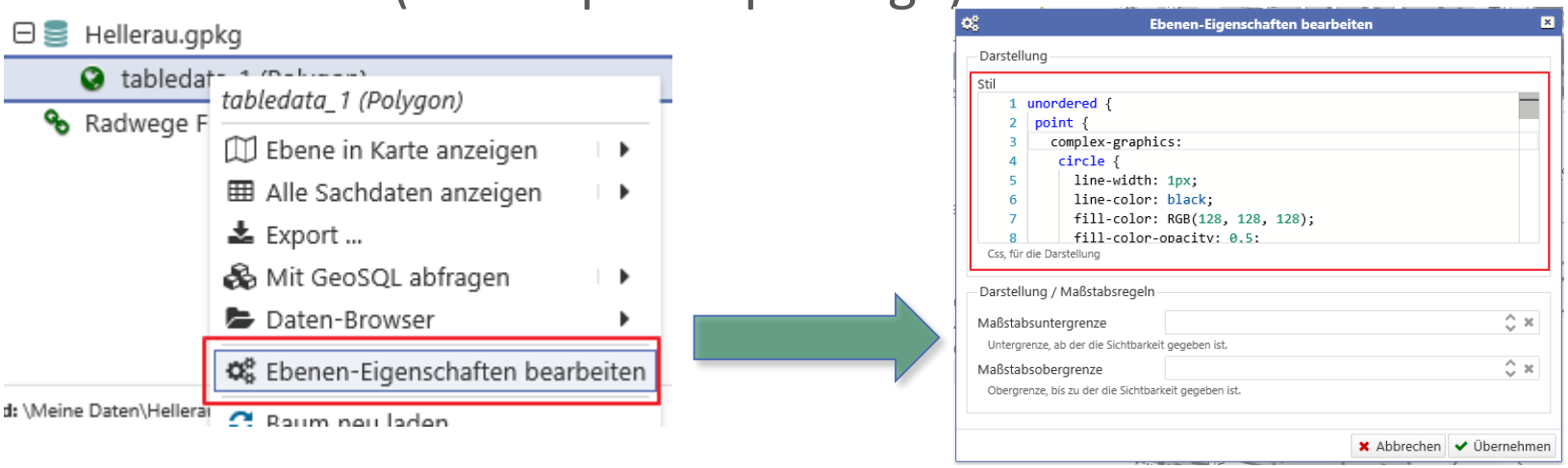
Hinterlegung des CSS im Managementcenter an der Ebene Gruppe – Anzeige /Darstellung

- **style**: der CSS-Code direkt (Beschränkung auf 4000 Zeichen)
- **cssFile**: Name einer Datei mit dem CSS-Code (der Inhalt wird in diesem Fall bei Änderungen automatisch neu gelesen),
- **fallbackStyle**: der CSS-Code direkt, dieser wird intern als "Notlösung" markiert, d.h. wenn die Datenquelle selber ein eigenes CSS mitbringt, wird das der Datenquelle bevorzugt verwendet (nur bei XPLAN Daten)

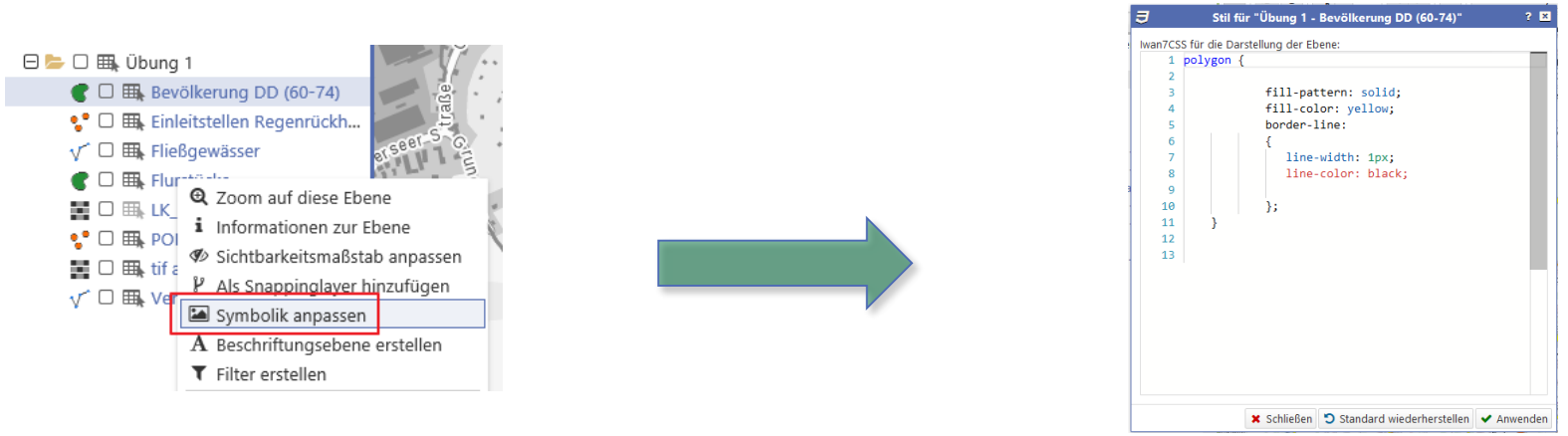
Sind alle Attribute angegeben, wird *cssFile* gegenüber *style* gegenüber *fallbackStyle* bevorzugt ausgewertet, d.h. effektiv kommt nur eine der Angaben zum Einsatz.

Einführung CSS

- CSS Anpassungen an der Datentabelle in der Anwendung Daten-Browser (hier Bsp. Geopackage)



- CSS Anpassungen an der Ebene im Themenbaum der Karte



Einführung CSS

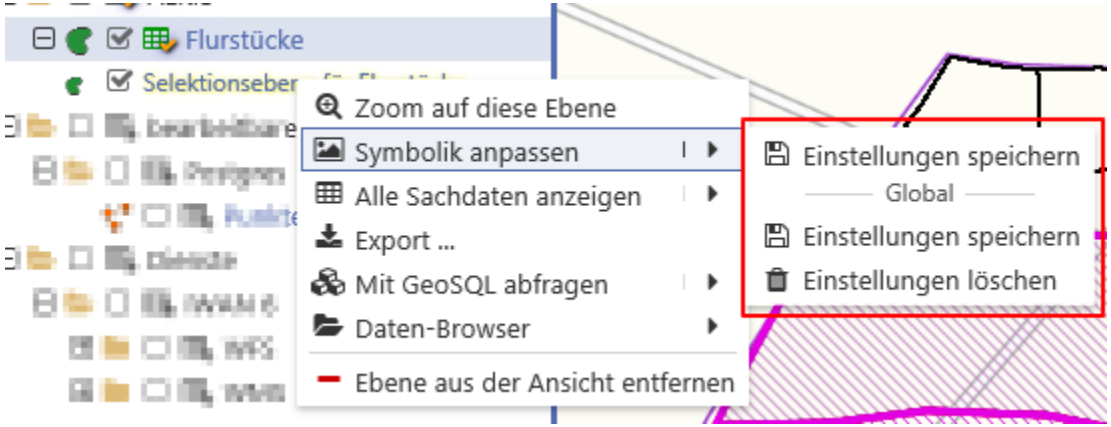
Daten-Browser

- Werden über dem Daten-Browser Themen zentral vorgegeben (Freigabe Datenverzeichnis), sind die adm. Anpassungen für alle Nutzer sichtbar
- Zu einer Datei kann eine gleichnamige CSS Datei für die Symbolik im gleichen Verzeichnis abgelegt werden. Diese wird bevorzugt verwendet.
- Bei Daten aus dem Daten-Browser: Anpassungen Symbolik ändern an der Ebene in der Karte, überschreiben CSS Vorgaben im Daten-Browser

Einführung CSS

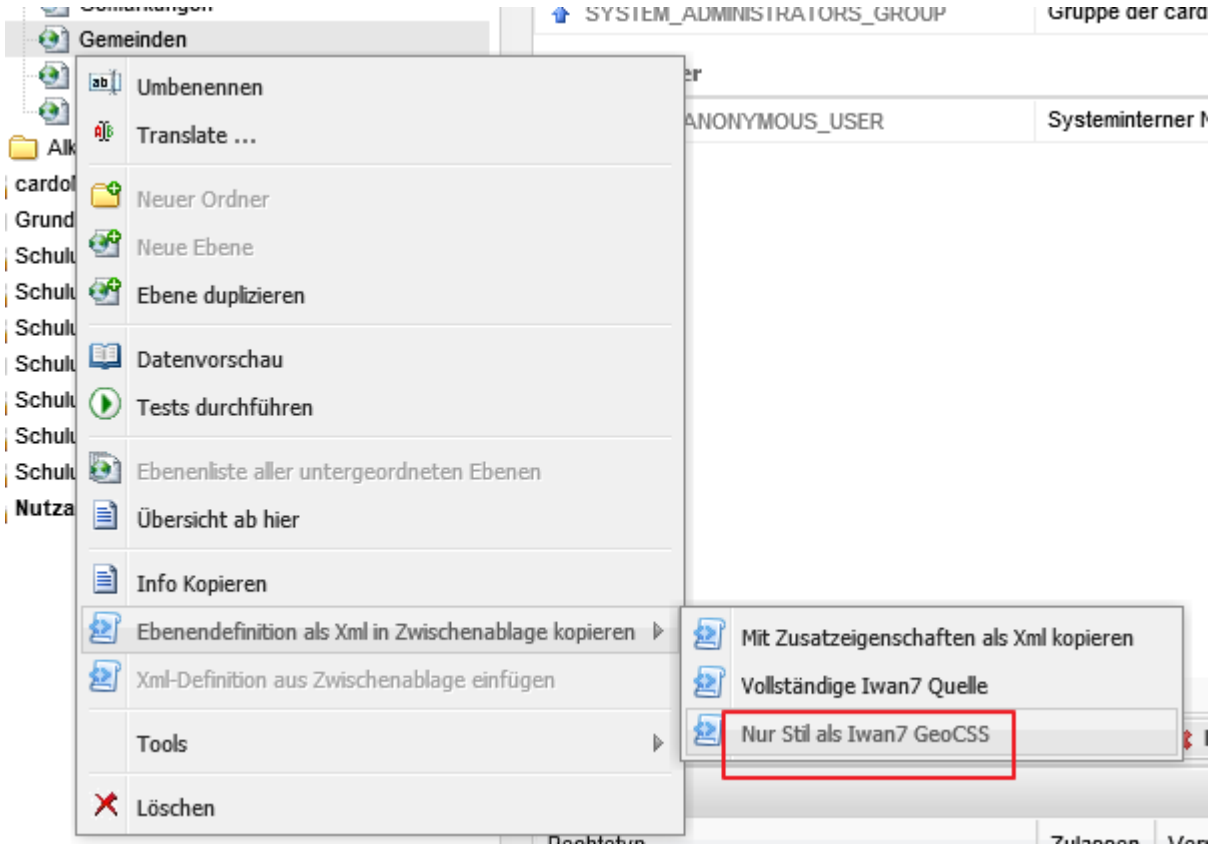
Symbolik ändern (Karte)

- Nicht für alle Ebenentypen möglich (DXF, WMS, …)
- Symbolikänderungen an der Karte werden in der Sitzung gespeichert
- Symbolikänderungen an der Karte -> kann jeder **Nutzer** für sich selber festlegen
- CSS Anpassungen für Selektionslayer: Administrative Einstellungen können global gespeichert werden



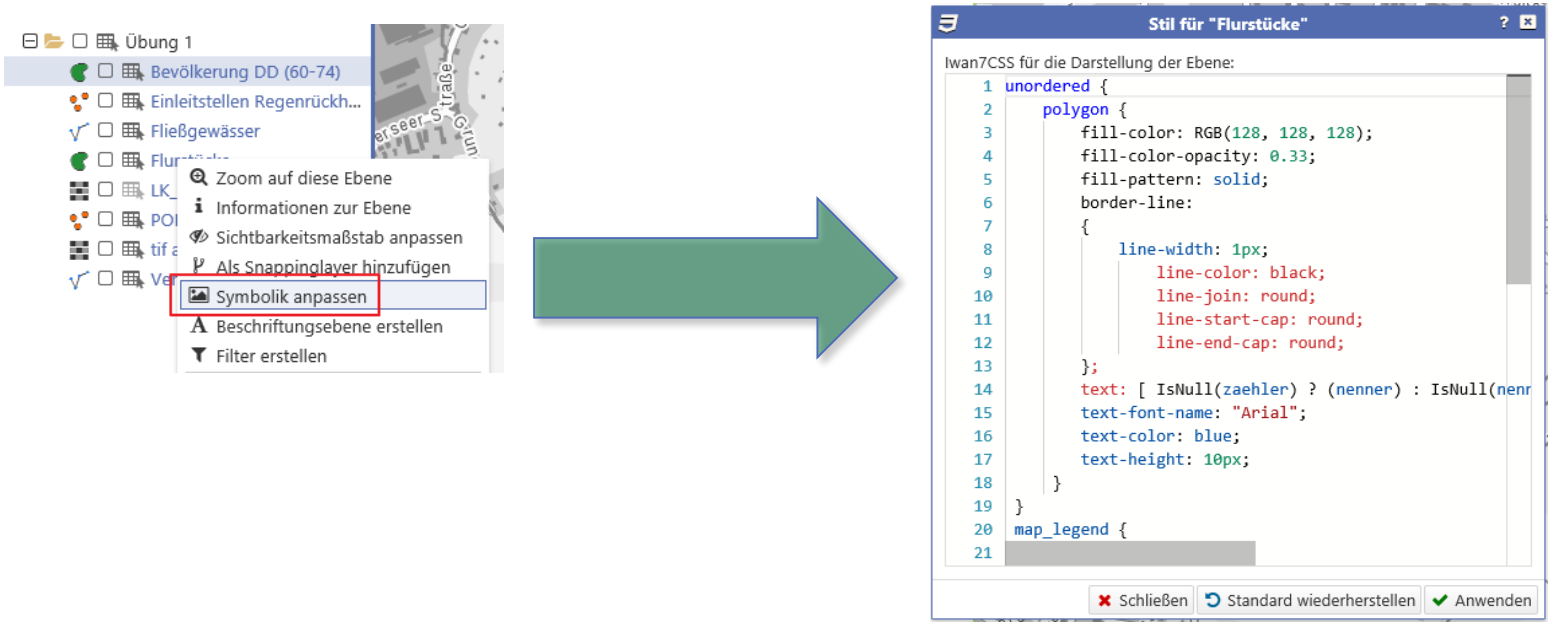
Einführung CSS

- Legendeneigenschaften können aus Iwan6 Ebenen als CSS konvertiert werden
- Nachbereitung teilweise erforderlich



Einführung CSS

- Ebenso Umwandlung der Symbolik einer Iwan6 Ebene als CSS beim Klick auf *Symbolik anpassen* im cardo4

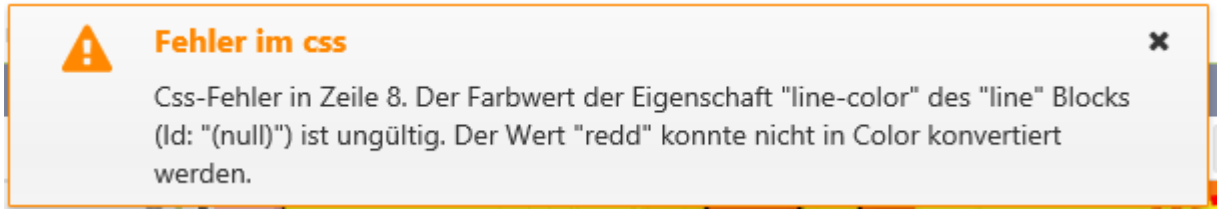


Arbeitsumgebung

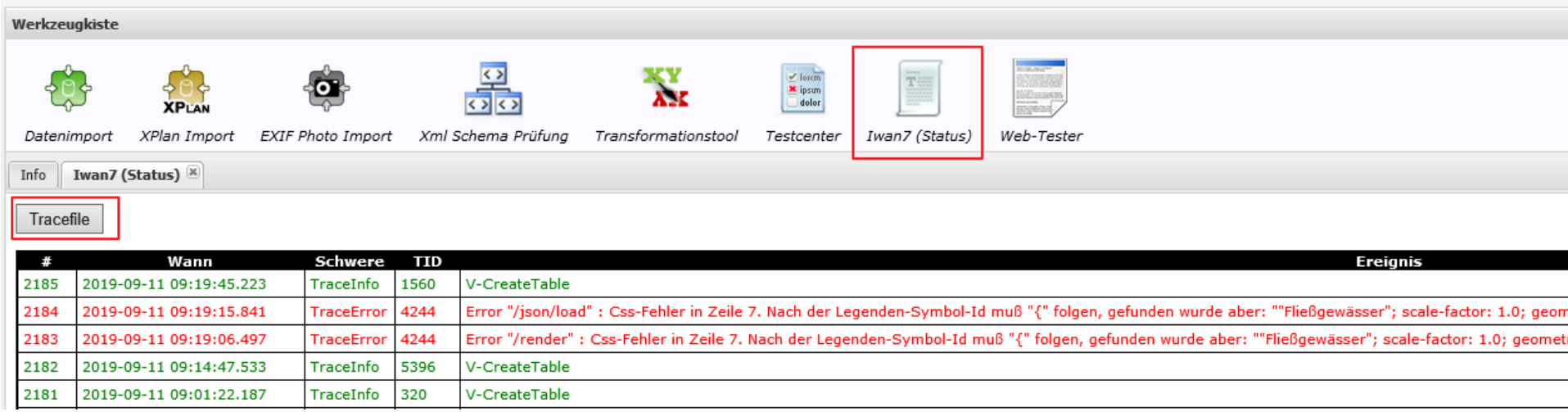
- Texteditor (Scite/Editor)
- Daten-Browser oder Symbolik anpassen am Datensatz mit Autovervollständigung und Beschreibung

Fehlermeldungen

- Direkt im Kartenbild



- Iwan7 Tracefile



Einführung CSS

- CSS Code aus Blöcken, in denen Eigenschaften Werte zugewiesen werden
- Block besteht aus geschweiften Klammern
- Blocktyp (line, point, polygon) bestimmt Eigenschaften

```
Blocktyp::BlockId {
    Eigenschaft1: Wert;
    Eigenschaft2: Wert;
}
```

- BlockId: frei wählbarer Name (wird bei Fehlermeldungen mit ausgegeben + für manuelle Erstellung einer Legende notwendig)
- BlockID darf kein Schlüsselwort sein

Einführung CSS

- BlockID muss eindeutig sein
- Mehrmalige Verwendung der BlockID führt zur Fehler-meldung
- Eigenschaftsname + Doppelpunkt + Wert + ;
- Pro Block jede Eigenschaft nur 1x

```
Blocktyp::BlockId {
    Eigenschaft1: Wert;
    Eigenschaft2: Wert;
}
```

```
line::Hauptstraße {
    line-width: 2;
}
```

Einführung CSS

- Blöcke können geschachtelt werden
- Blocktypen werden vererbt
- äußere Blocktypen dürfen keine Eigenschaften definieren
- Innere Blöcke erben den Blocktyp des Äußeren (der Innere darf selber keinen Blocktyp definieren)

```
line {
    Hauptstraße {
        line-width: 2;
    }

    Nebenstraße {
        line-width: 5;
    }
}
```

Einführung CSS

- Kommentare können einzeilig oder mehrzeilig sein

```
line::Hauptstraße {
    line-width: 2; // Dies ist ein einzeiliger Kommentar
}

/*
Dies ist ein mehrzeiliger Kommentar
*/
```

Farbangaben

- Farbangaben: engl. Farbangabe, RGB Farbcode, ARGB, #RGB
- Farbkanal 0…255, Alpha 0…1
- Siehe https://www.cardogis.com/Default.aspx?pgId=1258
- **Hinweis**: Standardfarbe für alle Geometrietypen ist schwarz.

Beispiel: Möglichkeiten – Farbzuweisung grüne Linie

- line-color: rgb(0,255,0); //Farbwert als RGB – Angabe
- line-color: ARGB(0.5, 0, 255,0); //Farbwert – Opazität 50%, RGB
- line-color: green; //Farbwert als Farbname
- line-color: green(0.8); //Farbwert als Farbname mit Transparenz
- line-color: #00ff00; //Hexadezimal ohne Transparenz
- line-color: #00ff00**FF**; //Hexadezimal – Opazität* der Fläche

*Opazität => Lichtundurchlässigkeit

CSS Linie

```
line {
  //Standardwert
  render-quality: antialiased;
  //Linienbreite in px oder m
  line-width: 2;
  //line-width: 2px;
  //line-width: 2m;
//Linienfarbe (Standard schwarz)
  line-color: RGB(0, 143, 255);
}
```

- Render-Qualität: aliased / antialised

- Mit LinienID

Blocktyp::BlockID

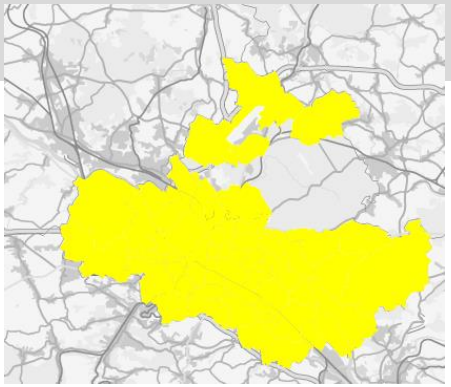
```
Line::LinienID {
   line-width: 2;
   line-color: RGB(0, 143, 255);
}
```

```
Line::Fließgewässer {
  line-width: 2;
  line-color: RGB(0, 143, 255);
}
```

CSS Polygon

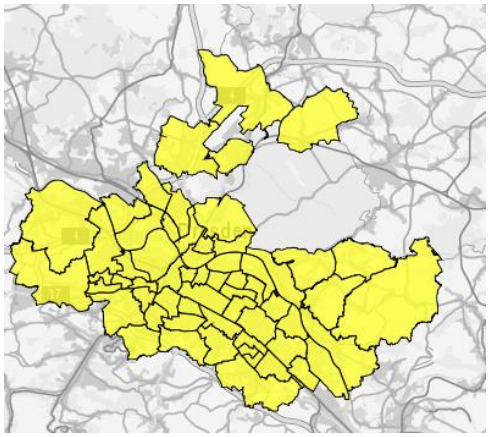
- Füllung (color = Farbe und pattern = Füllart)

```
polygon {
  //fill-color: yellow(0.5);
  fill-color: yellow;
  fill-pattern: solid;
}
```



CSS Polygon

- Rand des Polygons (border-line = Linie um das Polygon)
- Border-line ist wiederum eine (**komplexe**) Eigenschaft, definiert keine BlockID (ist auch kein Block)
- definiert selber Werte, die von der Eigenschaft abhängen

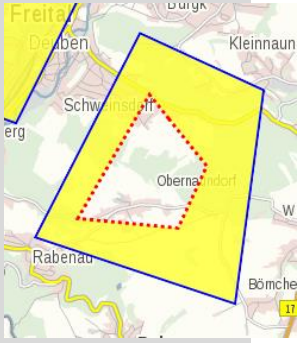


```
polygon {
  fill-color: yellow;
  fill-pattern: solid;
 //Transparenz 0 – 1, 1 = 100% Dreckkraft
  fill-color-opacity: 0.8;
  border-line:
            {
            line-width: 1px;
            line-color: green;
            };
}
```

CSS Polygon

- Outer-border-line, inner-border-line und border-line im Polygon
- Innerhalb des Blockes sind alle Eigenschaften erlaubt, die auch für line zulässig sind
- Abschluss des line-Blocks mit Semikolon

```
polygon {
 fill-color: yellow;
 fill-pattern: solid;
 fill-color-opacity: 0.8;
//Außenlinie
   outer-border-line:    {
      line-width: 2px;
      line-color: blue;
   };
//Innenlinie
   inner-border-line:    {
      line-width: 3px;
      line-color: red;
      line-dash-style: dot;
   };
}
```



CSS Punkt

```
point {
  complex-graphics:
   circle {
     line-width: 2px;
     line-color: red;
     fill-color: green;
     fill-color-opacity: 0.8;
     fill-pattern: solid;
     radius: 10px;
  };
}
```

- Punkt wird als komplexe Graphik definiert
- Punkt ist Kreis mit Füllfarbe
- Größe des Punktes über Radius

```
point {
  complex-graphics:
   circle {
              fill-color: green;
              fill-pattern: solid;
              radius: 10px;
  };
}
```

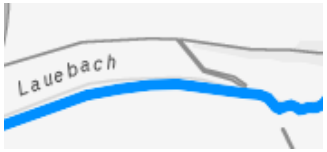
Datenvorbereitung

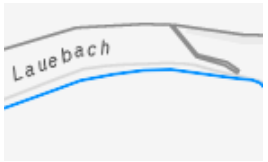
Übung 1

- CSS für eine Linie
- CSS für einen Punkt
- CSS für eine Fläche erstellen

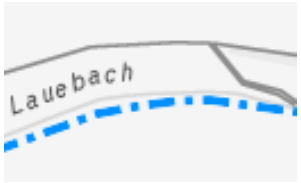
Linien

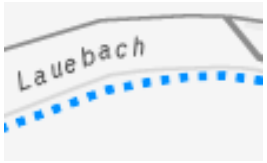
- Linienbreite
  - line-width: Physikalische Größe in m oder px (m muss als Einheit notiert werden) oder Ausdruck in Pixel [MeterToPixel(6.0)+4];





- Linie mit vordefiniertem Muster
  - Eigenschaft: line-dash-style (solid, dash, dot, dash_dot, dash_dot_dot, custom)

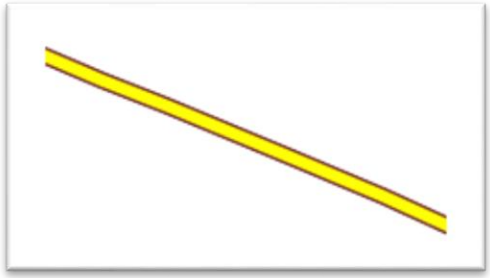




- Linienenden
  - Eigenschaft: line-cap (flat, square, round)

Linien

- Linie mit Straßensignatur
  - Mehrere (Linien)BLÖCKE werden übereinander gezeichnet



```
line {
   Landstraße1 {
            //6 m + 4 Pixel
            line-width: [MeterToPixel(6.0)+4];
            line-min-width: 2;
            line-max-width: 16;
            line-color: RGB(112, 39, 42);
   }
   Landstraße2 {
            line-width: 7.0m;
            line-min-width: 2;
            line-max-width: 14;
            line-color: RGB(255, 255, 0);
   }
}
```

Min-Max-Breiten bei Linien

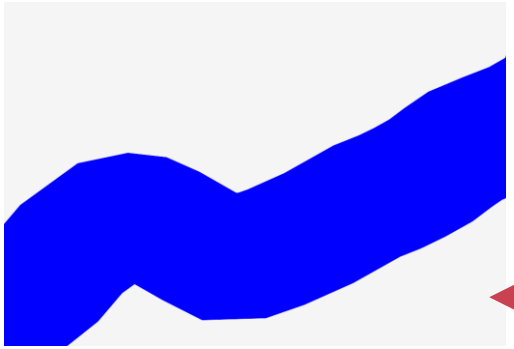
- line-width: Linienstärke in m (oder px)
- line-min-width: Mindestdicke der Linie in Pixel
- line-max-width: Maximaldicke der Linie in Pixel

Ohne Min-Max-Angabe

Mit Min-Max-Angabe

Maßstab 1:60 000

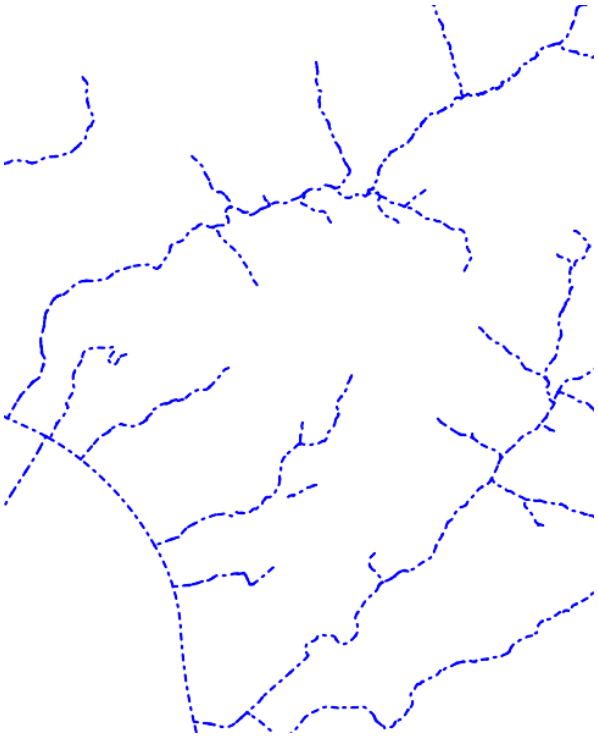
```
line {
  line-width: 10m;
  /*immer in Pixel (keine
Einheit notwendig) */
  line-min-width: 2;
  line-max-width: 5;
  line-color: blue;
}
```

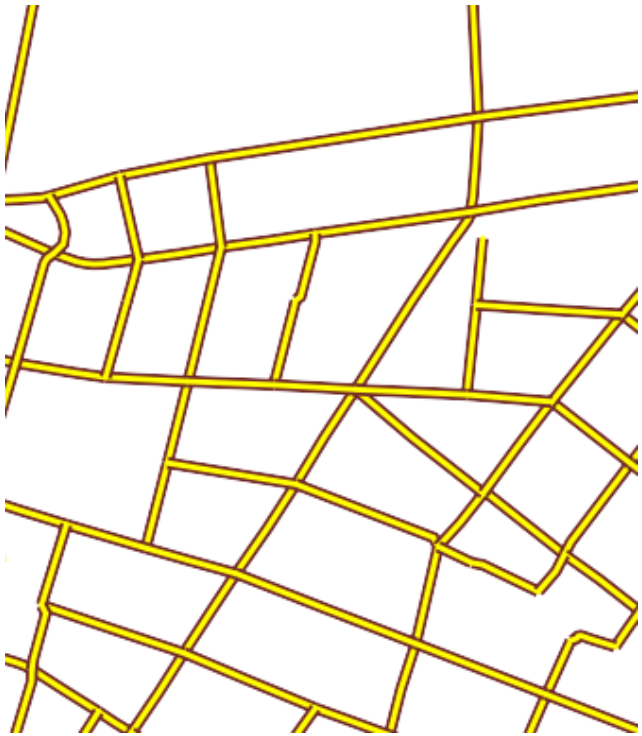


Maßstab 1:100

Übung 1a + 1b

- Gestrichelte Linie erstellen
- Linie mit Straßensignatur erstellen

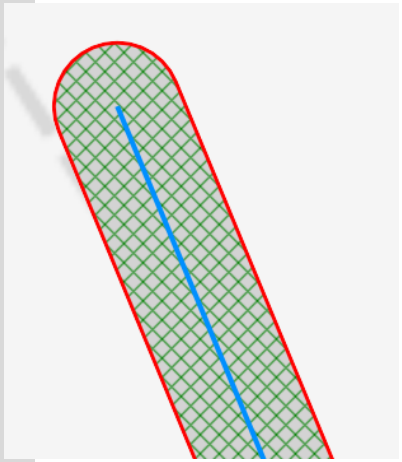




Puffer

```
line {
  line-color: RGB(0, 143, 255);
  line-width: 4px;
  buffer-distance: 50px;
  buffer-polygon: {
      fill-color: green;
      background-color: lightgrey;
      fill-pattern: hatch_style_diagonal_cross;
      pattern-lines-width: 2px;
      distance-between-pattern-lines: 10px;
      border-line: {
        line-color: red;
        line-width: 3px;
       };
    };
}
```

- Linie: Puffer-Polygon muss definiert werden
- *Fill-color* für das Füllmuster
- *Background-color* für den Hintergrund

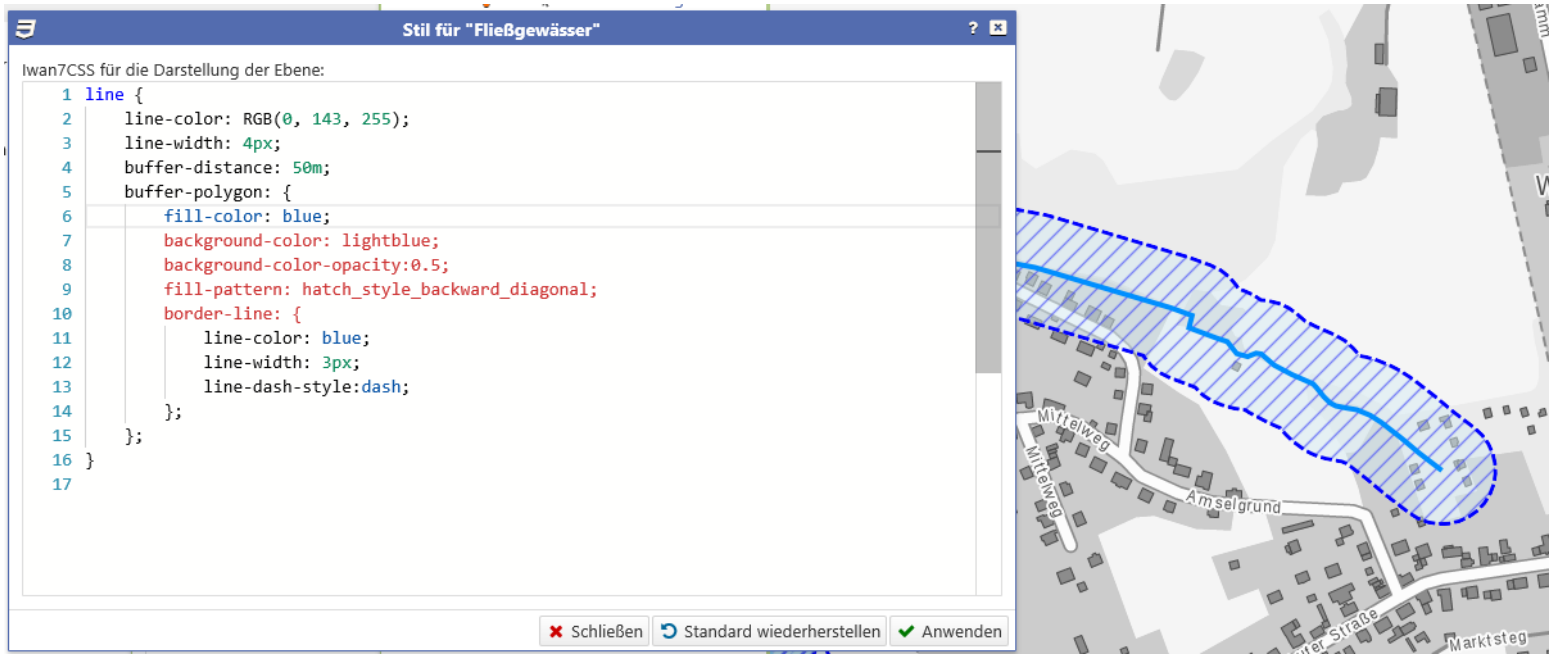


Puffer

```
polygon {
   fill-color: yellow;
   fill-pattern: solid;
   fill-color-opacity: 0.8;
   border-line:  {
                  line-width: 1px;
                  line-color: black;
                  };
    buffer-distance: 100m;
    buffer-polygon: {
       fill-color:red;
       fill-pattern: solid;
       fill-color-opacity:0.5;
       border-line: {
          line-color: red;
          line-dash-style: dot;
          line-width: 5px;
       };
   };
}
```

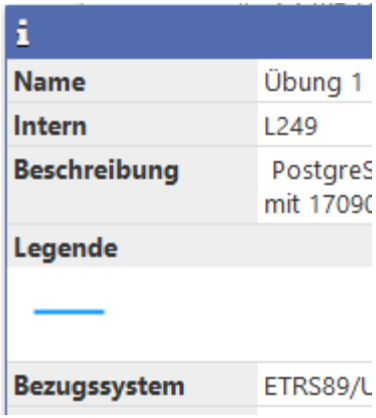
Übung 1c

- Gepufferte Linie erstellen

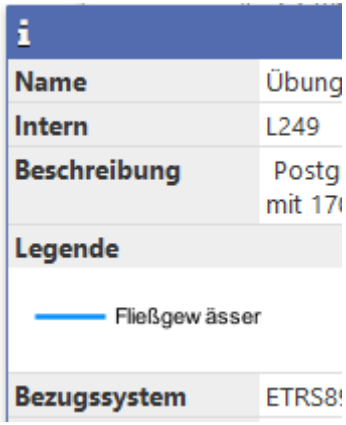


Automatisch generierte Legende

- Wird erstellt, wenn kein map_legend Eintrag vorhanden ist
- Ausschluss aus Legende mittels exclude-from-map-legend: true;
- Bedingungen und Eigenschaften von CSS Blöcken dürfen keine dynamische Funktion enthalten (Spaltenwert * 2)
- Erlaubt sind zwei optionale Eigenschaften:
  - Map-legend-label
  - Map-legend-scale-factor: Skalierung der graphischen Elemente des Symbols



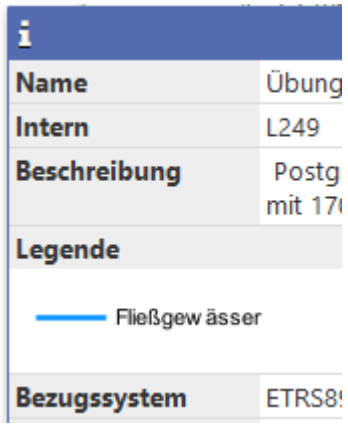

```
line {
        line-width: 2;
        line-color: RGB(0, 143, 255);
        map-legend-label: „Fließgewässer";
        map-legend-scale-factor: 1;
}
```



Legende

- Explizite Definition einer Legende
  - Immer außerhalb der Legendendefinition
  - Keine BlockID zu definieren

```
map_legend {
        {
        label: „Fließgewässer";
        scale-factor: 1.0;
        geometry-type: line;
        }
}
```



Legende

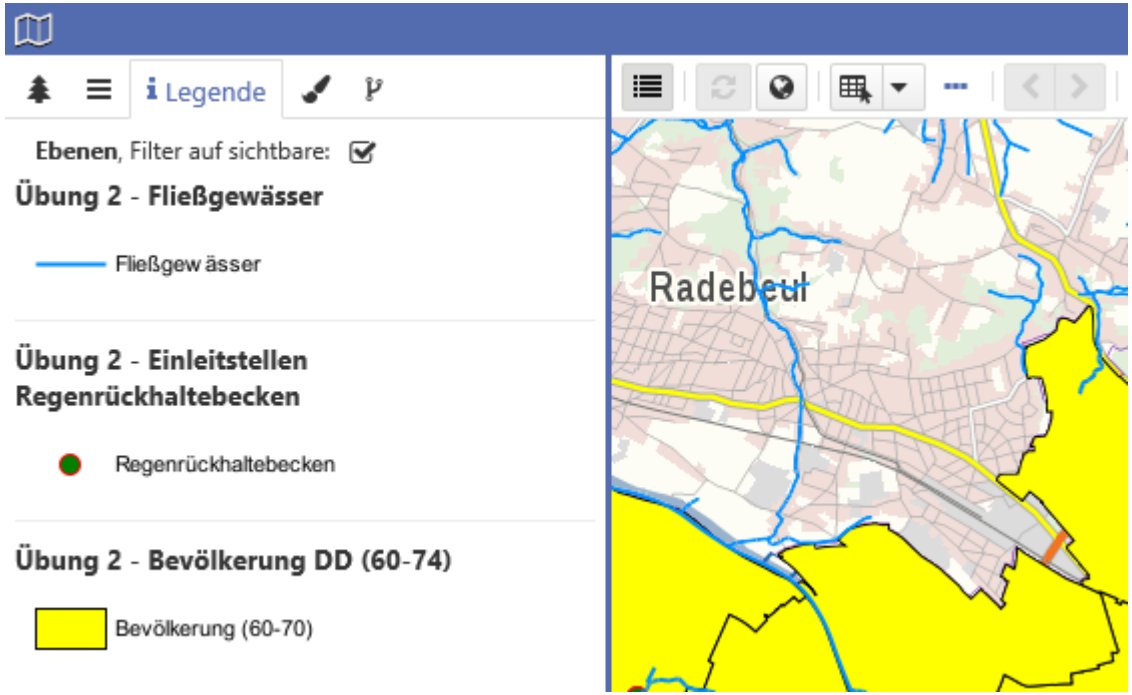
- Überprüfung von Legenden mittels URL-Aufruf

- http://localhost:8287/iwan/legend?cssFile=D:/Dev/cardoSystem/_Projekt_xy/Geodaten/css/uebung1_bevoelkerung.css

- http://zid2:8287/iwan/legend?cssFile=d:/Dev/cardoSystem/_Projekt_SCHULUNG_Dozent/Geodaten/CSS/Schulung/uebung1_bevoelkerung.css

Übung 2

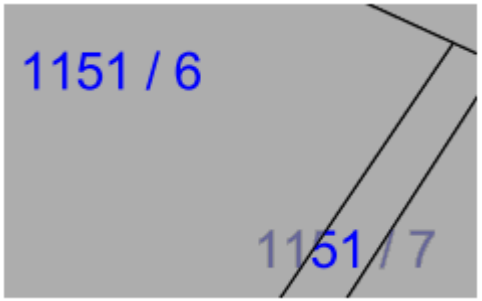
- Legende für die in Übung 1 erstellten Ebenen hinterlegen

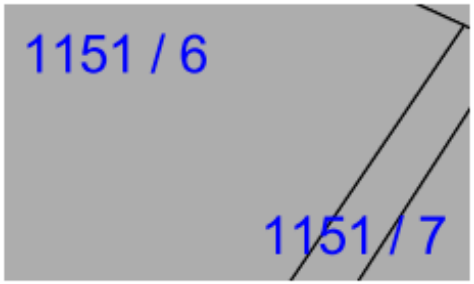


Zeichenreihenfolge

- ordered / unordered
- Angabe vor dem Geometrietyp
- Bestimmt die Zeichnungsreihenfolge der Blöcke

```
ordered {
   polygon {
      fill-color: RGB(128, 128, 128);
      fill-color-opacity: 0.8;
      fill-pattern: solid;
      border-line: {
         line-width: 1px;
         line-color: black;
         line-join: round; };
   }
   polygon [mapscale <= 3000] {
      text: [(zaehler) // " / " // (nenner) ];
      text-font-name: "Arial";
      text-color: blue;
      text-height: 20px;
   }
}
```





Defaultwerte

- Festlegung von voreingestellten Werten (bspw. Farbe oder Linienbreite etc.), bilden den Standard für diese Legende
- So lange gültig, bis ein anderer Default-Wert oder Wert diese Definition überschreibt

- Default steht am Anfang des CSS und muss nach dem Geometrietyp folgen
- Es darf keine BlockID notiert werden
- Default-Wert allein zeichnet keine Geometrie

```
line::default {
   line-color: red;
}
```

Bedingungen

- Müssen erfüllt werden, damit ein Block gezeichnet wird
- Werden in [] Klammern geschrieben
- Variablen sind hierbei Spaltennamen der Datenquelle
- Zeichenketten müssen in doppelten “ geschrieben werden

```
ordered {
   line {
      Hauptstraße [Geotyp == "HS"] {
         line-width: 2;
      }

      Landstraße [Geotyp == "LS"] {
         line-width: 5;
      }
   }
}
```

Operatoren

https://www.cardogis.com/Default.aspx?pgId=1839

Operator	Beschreibung
Mathematische Operatoren	
+	Addition (a + b)
-	Subtraktion (a - b)
*	Multiplikation (a * b)
/	Division (a / b)
^	Potenz (a ^ b)
Vergleiche	
==	Gleichheit (a == b)
!=	Ungleichheit (a != b)
>	größer als (a > b)
<	kleiner als (a < b)
>=	größer oder gleich (a >= b)
<=	kleiner oder gleich (a <= b)
Vergleiche mit impliziter Typkonvertierung	
===	Gleichheit (a === b)
!==	Ungleichheit (a !== b)
>>>	größer als (a >>> b)
<<<	kleiner als (a <<< b)
>==	größer oder gleich (a >== b)
<==	kleiner oder gleich (a <== b)

[PARAM === "1000"]

Ist der Datentyp von PARAM eine Zahl wird die Zeichenkette "1000" in eine Zahl konvertiert und dann mit dem Wert aus PARAM auf Gleichheit getestet. Ist der Datentyp von PARAM hingegen eine Zeichenkette, werden die beiden Zeichenketten direkt auf Gleichheit getestet.

Operatoren

https://www.cardogis.com/Default.aspx?pgId=1839

Logische Operatoren	
&&	logisches und (a && b) -> a und b müssen erfüllt sein
||	logisches oder (a || b) -> a oder b müssen erfüllt sein
String Operationen	
//	Zeichenketten-Verknüpfungsoperator (a // b) -> Beispiel: [ABC // DEF] ergibt ABCDEF
Sonstige Operatoren	
?:	if (wenn) then (dann) else (sonst) Operator (a > b ? 10 : 20) -> das Ergebnis ist 10 wenn a größer als b sonst 20

Bedingungen

- Charakter = Charakter
- [Stadtteilname == "Hellerau/Wilschdorf mit Rähnitz"]

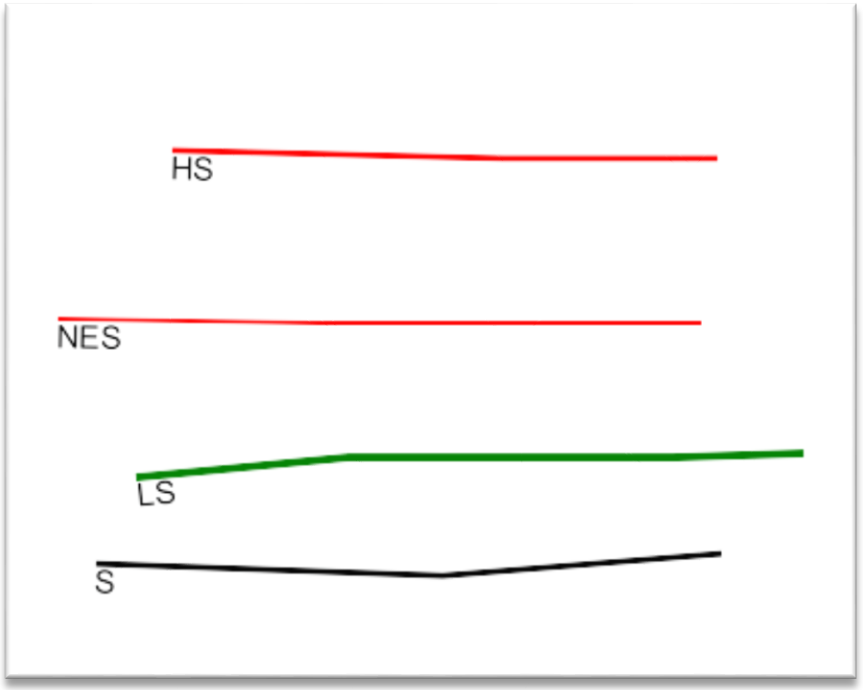
- Zahl = Zahl
- [prozent = 5]

- Charakter = Zahl
- [ordn == "2"]
- Implizite Typkonvertierung
- [ordn === 2]

Defaultwerte + Bedingung

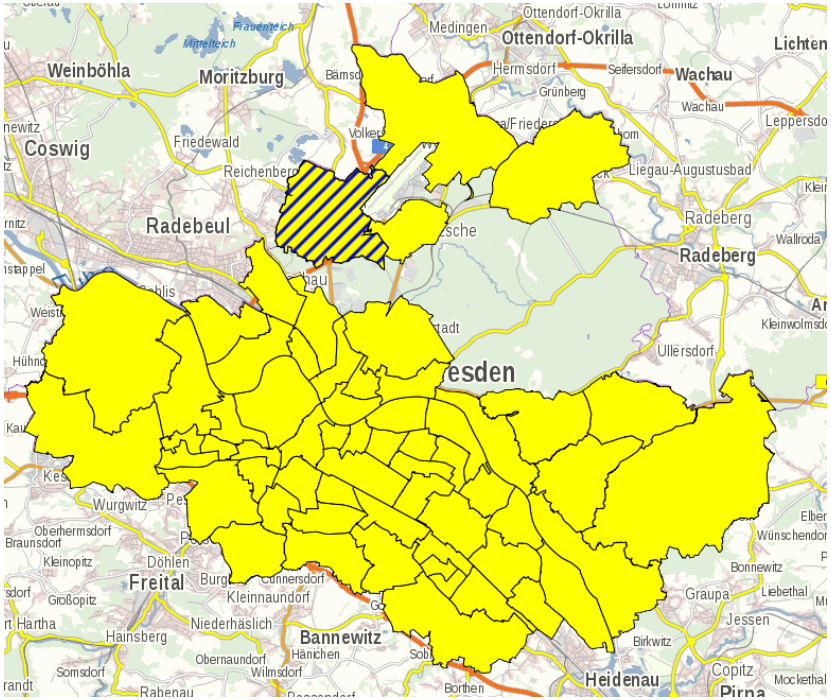
```
line::default {
   line-color: red;
}
unordered {
   line::Hauptstraße [Geotyp == "HS"] {
         line-width: 2;      }
    line::Nebenstraße [Geotyp == "NES"] {
         line-width: 1;       }
   line::Landstraße [Geotyp == "LS"] {
         line-width: 3;
         line-color: green;    }
}
line::default {
   line-color: black;
}

ordered::line::Schiene [Geotyp == "S"] {
   line-width: 3;
}
```



Übung 3

- Bedingung definieren [Stadtteilname == "Hellerau/Wilschdorf mit Rähnitz"]
- Defaultwerte nutzen
- Zeichnungsreihenfolge beachten



Bedingungen

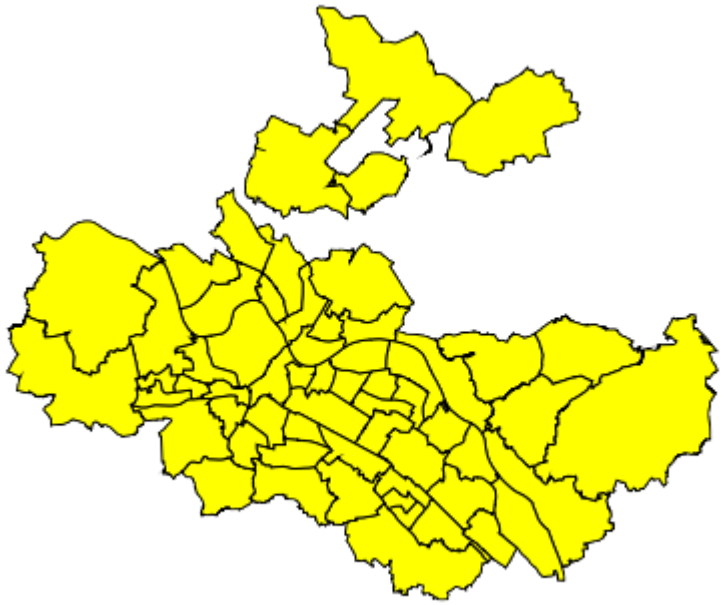
- Kombination von Bedingungen möglich
- Können geschachtelt werden
- Reihenfolge der Abarbeitung kann mit runden Klammern gesteuert werden
- Bedingung muss logischen Ausdruck ergeben, der wahr oder falsch sein muss
- Bspw. mit dem Maßstab (Variable mapscale oder mapscale6)

```
line {
      Hauptstraße_1 [Geotyp == "HS“ && mapscale6 <= 10000] {
        line-width: 2;
      }
      Hauptstraße_2 [(Geotyp == "HS") && (mapscale6 > 10000)] {
        line-width: 5;
      }
}
```

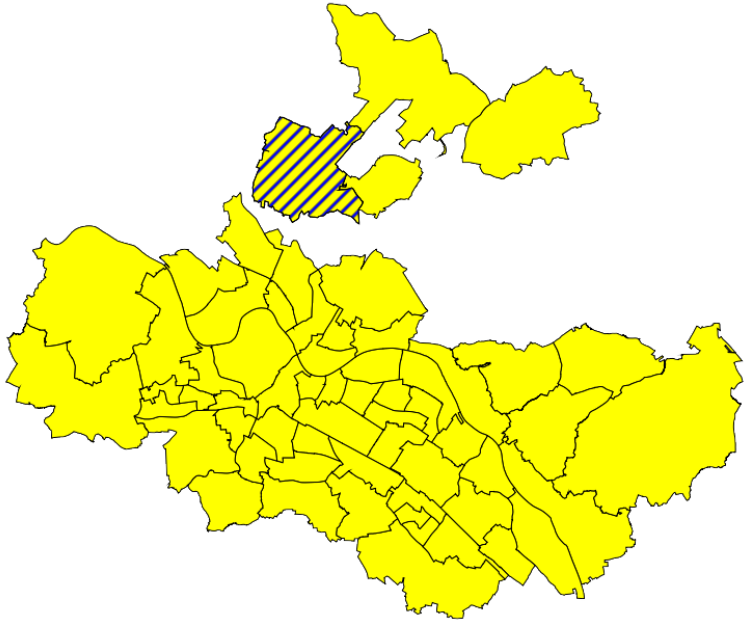
Übung 4

- Zusätzlich eine Maßstabsbeschränkung setzen

1:240000



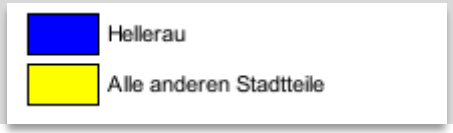
1:120000



Legende

- Explizite Definition einer Legende
  - Bei variablen Ausdrücken, müssen die Daten, die für ihre Auswertung notwendig sind, angegeben werden
  - Entweder über data Definition oder über geometry-IDs

```
map_legend {
        {
        label: „Hellerau";
        scale-factor: 1.0;
        geometry-type: polygon;
        data: stadtteilname = "Hellerau/Wilschdorf mit Rähnitz";
        }
        {
        label: "Alle anderen Stadtteile";
        scale-factor: 1.0;
        geometry-type: polygon;
        data: stadtteilname = "Alle anderen Stadtteile";
        }
}
```



Legende

- Angabe über BlockIDs (werden als geometry-id in der Legende referenziert)
- Nur nutzen, wenn keine dynamischen Funktionen verwendet werden

```
ordered {
polygon {
  AlleAnderen {
                fill-color: yellow;
  }
  Hellerau [(Stadtteilname == "Hellerau/Wilschdorf mit Rähnitz")
                && (mapscale <= 150000)] {
                 fill-color: blue;
                 fill-pattern: hatch_style_backward_diagonal;
                 pattern-lines-width: 3px;
                 background-color: white;
  }
}
}
```

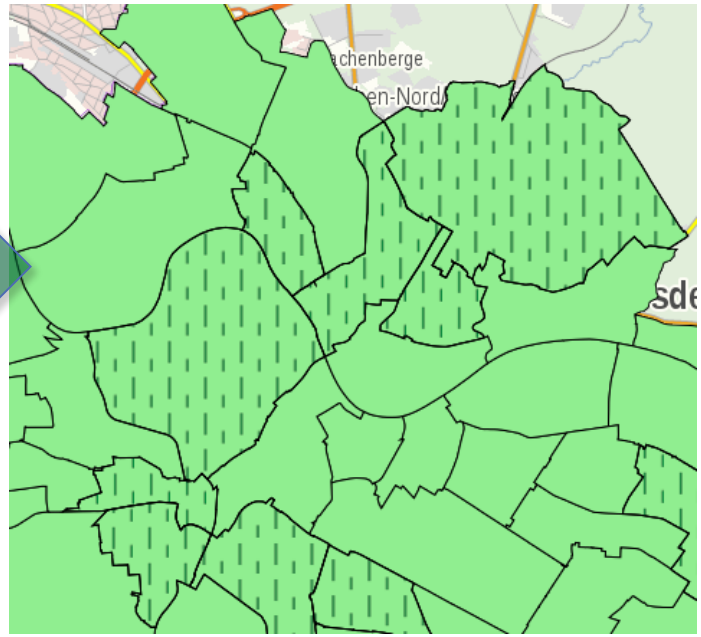
```
map_legend {
{
label: "Alle anderen Stadtteile";
scale-factor: 1.0;
geometry-type: polygon;
geometry-ids: alleanderen;
}

{
label: "Hellerau";
scale-factor: 1.0;
geometry-type: polygon;
geometry-ids: Hellerau;
}
}
```

Legende

- Angabe von mehreren Block Ids in einer Legendendefinition

```
polygon {
    Wald {
    fill-color: lightgreen;
    }
    Laubwald [prozent < 10] {
    fill-pattern: complex;
    complex-fill-pattern-bbox-width: 28;
    complex-fill-pattern-bbox-height: 28;
    complex-fill-pattern-width: 28m;
    complex-fill-pattern-min-width: 25;
    complex-fill-pattern-max-width: 100;
    complex-fill-pattern-height: 28m;
    complex-fill-pattern-min-height: 25;
    complex-fill-pattern-max-height: 100;
    complex-fill-pattern:
        text {
            text: "|";
```



Legende

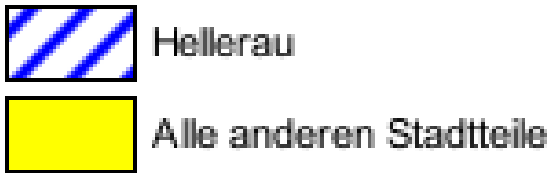
```
map_legend {
    {
        label: "Laubwald";
        scale-factor: 1.0;
        geometry-type: polygon;
        geometry-ids:  Laubwald;

        }
        {
        label: "Wald";
        scale-factor: 1.0;
        geometry-type: polygon;
        geometry-ids:  Wald;
        }
}
```


```
map_legend {
    {
        label: "Laubwald";
        scale-factor: 1.0;
        geometry-type: polygon;
        geometry-ids:  Wald, Laubwald;
        }
}
```

Übung 5

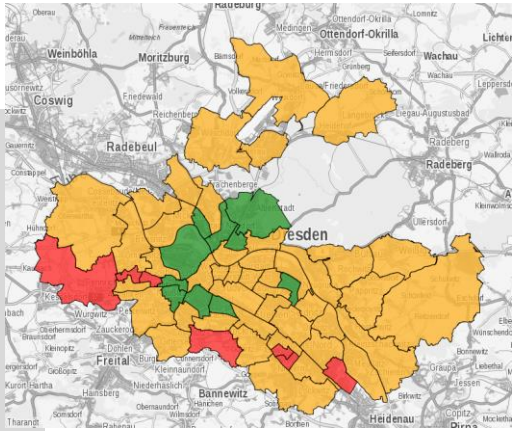
- Legende erstellen

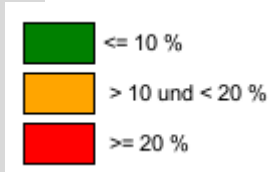


Klassifikationen

- Einfache Klassifikation nach einer Spalte

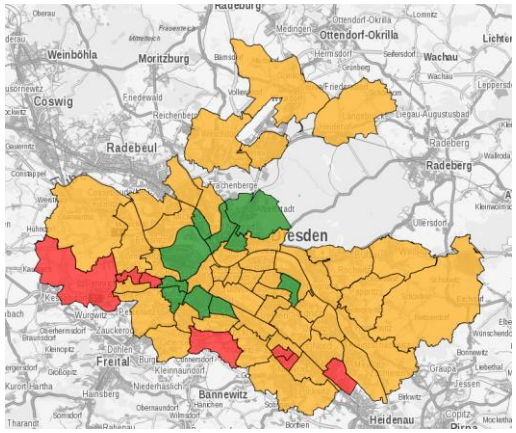
```
polygon {
        gruen [prozent <= 10] {
                fill-color: green;
        }
        orange [(prozent > 10) && (prozent < 20)] {
                fill-color: orange
        }
        red [(prozent >= 20)] {
                fill-color: red;
        }
}
```





Übung 6

- Klassifikation erstellen



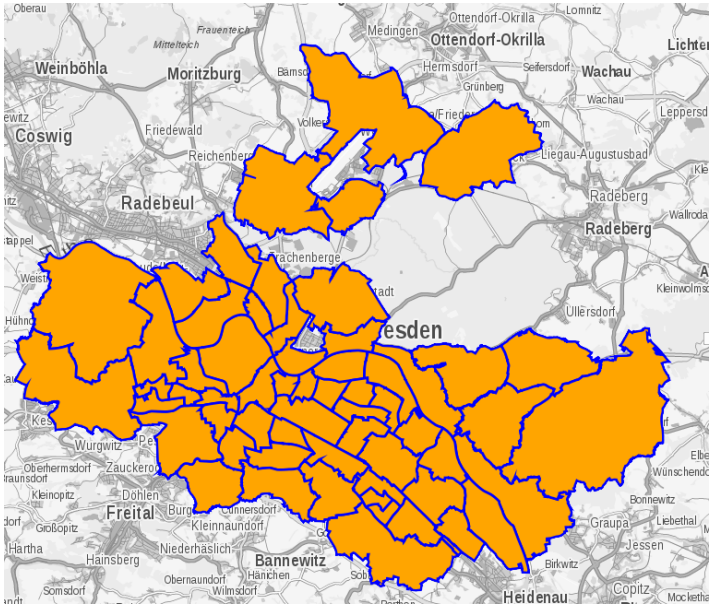
Optimierung von Bedingungen

- Ohne choice / Single-choice / Multi-choice
- Stellt sicher, dass nur ein Block gezeichnet wird, wenn die Bedingung zutrifft (auch wenn mehrere vorhanden sind)
- Single-choice: zeichnet nächsten Datensatz, sobald die Bedingung erfüllt ist (auch wenn Block leer ist und nichts gezeichnet wurde)
- Multi-choice: geht alle Blöcke durch, deren Bedingung zutrifft und fährt fort, sobald tatsächlich was gezeichnet wurde.
- Choice: Darf keine Blockid definieren und enthält keine Eigenschaften

choice

- ohne choice-Angabe

```
polygon::default {
        fill-pattern: solid;
        border-line: {
                line-width: 1px;line-color: black;
        };
}
polygon {
    gruen [prozent > 5 ] {
        [mapscale > 100000]{
                fill-color: green;
        }
      }
    orange [prozent > 5] {
        fill-color: orange;
     }
}
```

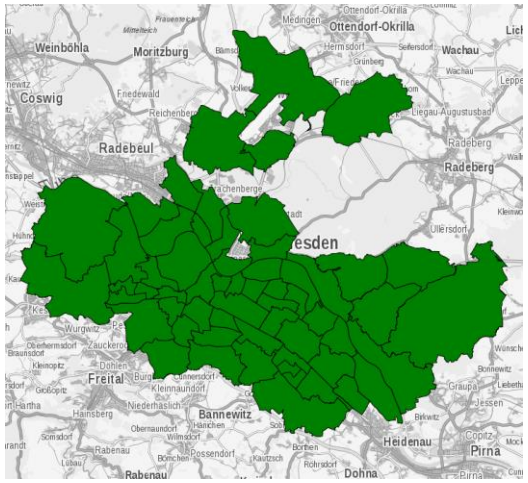


Arbeitet die Blöcke der Reihe nach ab, zeichnet grünen Block und danach darüber den orangen Block.

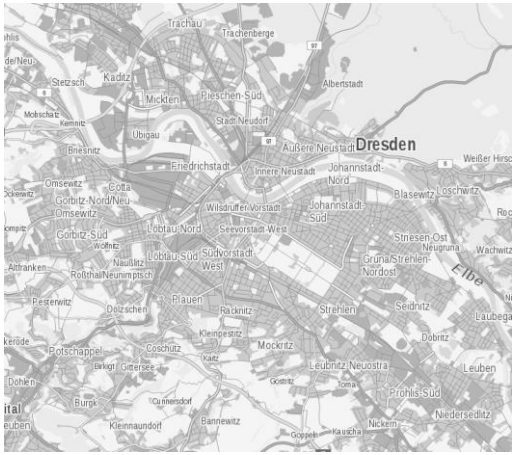
choice

- single-choice (Eine Auswahl)

```
...
polygon {
single-choice {
gruen [prozent > 5] {
            [mapscale > 100000]{
                        fill-color: green;
            }
   }
orange [prozent > 5] {
            fill-color: orange;
            border-line: {
                        line-color: blue;
                        line-width:100m;
      };
   }
}
}
```



1:120.000



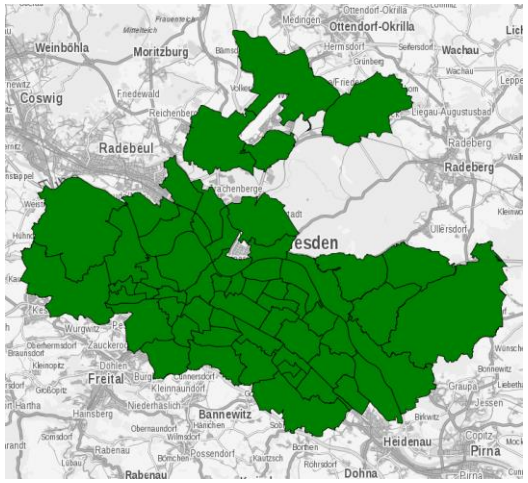
1:60.000

- Grüner Block wird nur ausgeführt

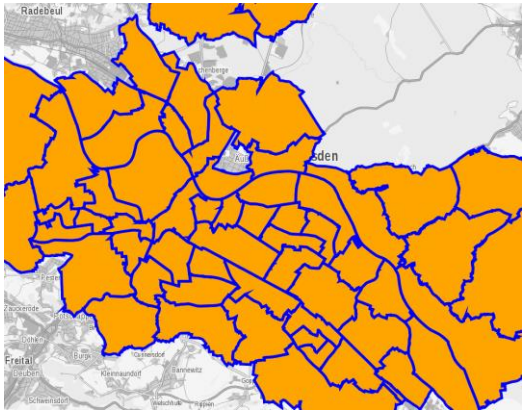
choice

- multi-choice

```
...
polygon {
multi-choice {
gruen [prozent > 5] {
                [mapscale > 100000]{
                                fill-color: green;
                }
   }
orange [prozent > 5] {
                fill-color: orange;
                border-line: {
                                line-color: blue;
                                line-width:100m;
      };
   }
}
}
```



1:120.000



1:60.000

- Zeichnet grünen Block wenn beide Bedingungen erfüllt sind.
- Außerhalb des MS-Bereiches wird oranger Block gezeichnet.

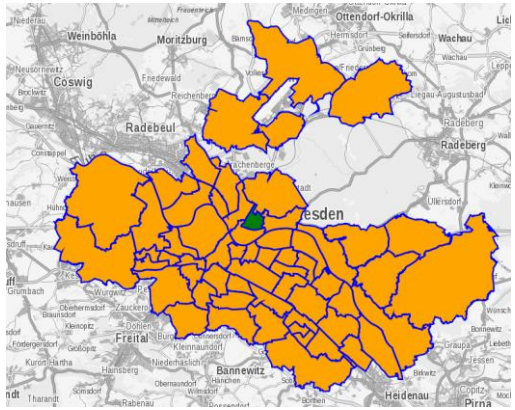
choice

kein choice	single-choice	multi-choice
Zeichenop1	Zeichenop1	Zeichenop1
und Zeichenop2	**oder** Zeichenop2	**oderWenn1leer** Zeichenop2
und Zeichenop3	**oder** Zeichenop3	**oderWenn2leer** Zeichenop3

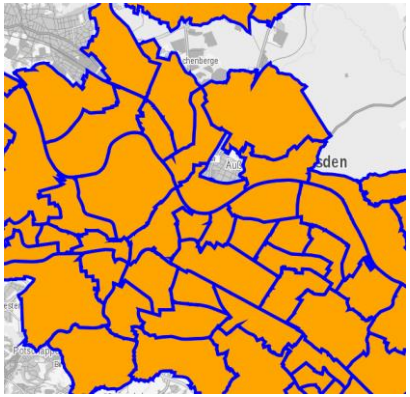
choice

- Bedingung anders definiert

```
...
polygon {
single-choice {
gruen [prozent < 5] {
                [mapscale > 100000]{
                                fill-color: green;
                }
   }
orange [prozent > 5] {
                fill-color: orange;
                border-line: {
                                line-color: blue;
                                line-width:100m;
        };
   }
}
}
```



1:120.000



1:60.000

- Single + Multi-Choice + ohne -> identisches Ergebnis
- Bedingungen überschneiden sich nicht

Übung 7a

- Choice Abfragen testen

Optimierung von Bedingungen

- Erzeugung eines Filters
- Es werden nur die Datensätze abgerufen, die auch gezeichnet werden sollen
- Erstellung des Filters geschieht automatisch

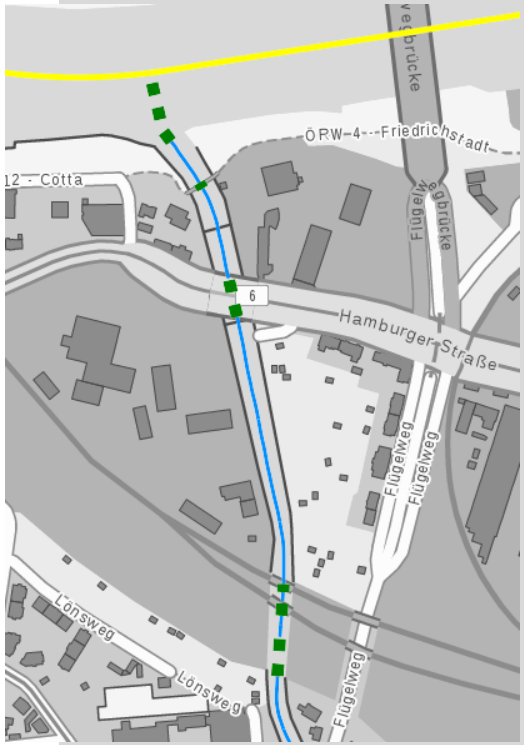
```
Unordered::line {
                    single-choice  {
                                settings {
                                            allow-query-optimization: true;
                                }
                                Hauptstraße1 [Geotyp == "HS"]{
                                            line-width: 2;          }

                                Landstraße1 [Geotyp == "LS"] {
                                            line-width: 5;          }
}}
```

Bedingung (Klassifikationen)

- Klassifikation nach zwei Spalten

```
… ordered {
        line {
                single-choice {
                        Elbe [(gn == "Elbe") && (rohr == 0)] {
                                line-width: 5px;
                                line-color: yellow;
                        }
                        offen [(rohr == 0) || (ordn === 2) ] {
                                line-width: 3px;
                        }
                        verrohrtkünstlich [(rohr == 1) && (ordn == "K")]{
                                line-dash-style: dot;
                                line-color: red;
                                line-width: 10px;
                        } …
```



Übung 7

- Klassifikation Fließgewässer nach mehreren Spalten

Vgl. single-choice vs. ohne choice

Stil für "Übung 7 - Fließgewässer"

lwan7CSS für die Darstellung der Ebene:

```
line::default {
    render-quality: antialiased;
    line-color: RGB(0, 143, 255);
    line-width: 2px;

}
line {
        single-choice {
            Elbe [(gn == "Elbe") && (rohr == 0)] {
                line-width: 5px;
                line-color: yellow;
            }
            offen [(rohr == 0) || (ordn === 2) ] {
                line-width: 3px;
            }

            verrohrtkünstlich [(rohr == 1) && (ordn == "K")]{
                line-dash-style: dot;
                line-color: red;
                line-width: 10px;

            }
            verrohrt [(rohr === 1) && (ordn === 0) || (ordn === 2) ]{
                line-dash-style: dot;
                line-color: orange;
                line-width: 10px;
            }
            rest    [(ordn === 1) ] {
                line-color: green;
                line-dash-style: dot;
                line-width: 10px;
                }

        }
    }
map_legend {
    {
    label: "Elbe";
    scale-factor: 1.0;
    geometry-type: line;
    data: gn="Elbe", rohr = 0, ordn = 0;
    }
    {
    label: "offen";
    scale-factor: 1.0;
    geometry-type: line;
    data: gn="andere Gewässer", rohr = 0, ordn = 2;
    }
    {
    label: "verrohrt künstlich";
    scale-factor: 1.0;
    geometry-type: line;
```

Schließen | Standard w

Karte

Legende

Ebenen, Filter auf sichtbare:

Übung 7 - Fließgewässer

- Elbe
- offen
- verrohrt künstlich
- verrohrt
- rest

Messwerkzeug

EPSG:25833 1:122299

Vgl. single-choice vs. ohne choice

Stil für "Übung 7 - Fließgewässer"

Iwan7CSS für die Darstellung der Ebene:

```
line::default {
    render-quality: antialiased;
    line-color: RGB(0, 143, 255);
    line-width: 2px;

}
line {
    //  single-choice {
            Elbe [(gn == "Elbe") && (rohr == 0)] {
                line-width: 5px;
                line-color: yellow;
            }
            offen [(rohr == 0) || (ordn === 2) ] {
                line-width: 3px;
            }

            verrohrtkünstlich [(rohr == 1) && (ordn == "K")]{
                line-dash-style: dot;
                line-color: red;
                line-width: 10px;

            }
            verrohrt [(rohr === 1) && (ordn === 0) || (ordn === 2) ]{
                line-dash-style: dot;
                line-color: orange;
                line-width: 10px;
            }
            rest    [(ordn === 1) ] {
                line-color: green;
                line-dash-style: dot;
                line-width: 10px;
                }

        //}
    }
map_legend {
    {
    label: "Elbe";
    scale-factor: 1.0;
    geometry-type: line;
    data: gn="Elbe", rohr = 0, ordn = 0;
    }
    {
    label: "offen";
    scale-factor: 1.0;
    geometry-type: line;
    data: gn="andere Gewässer", rohr = 0, ordn = 2;
    }
    {
    label: "verrohrt künstlich";
    scale-factor: 1.0;
    geometry-type: line;
```

Schließen | Standard w

Karte

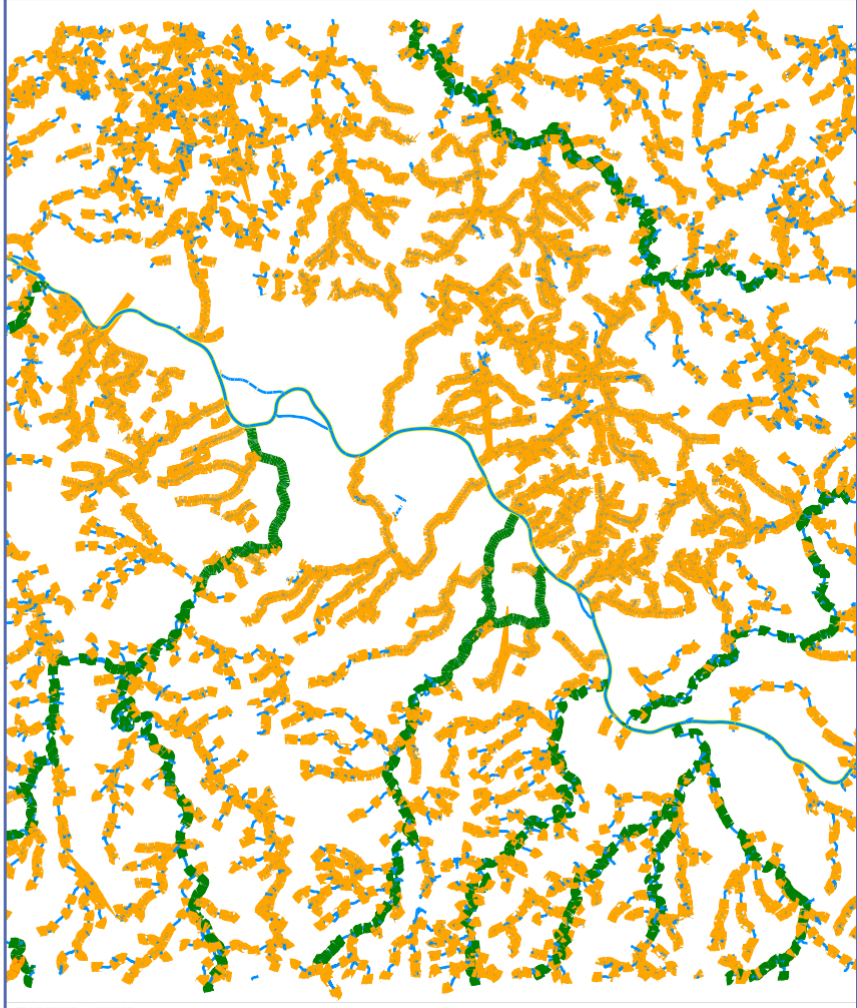
Legende

Ebenen, Filter auf sichtbare:

Übung 7 - Fließgewässer

- Elbe
- offen
- verrohrt künstlich
- verrohrt
- rest

Messwerkzeug

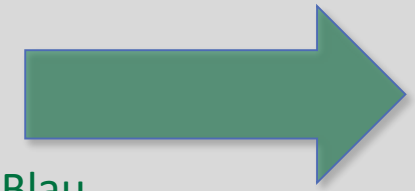


EPSG:25833 1:122299

Ausdrücke in Eigenschaftszuweisungen

- In [] Klammern nach Eigenschaftsnamen und Doppelpunkt
- Ersetzen die direkte Wertzuweisung
- Bspw. Werte aus der Datenquelle für Styles
- Ergebnis eines Ausdrucks muss vom Typ der Eigenschaft sein
- Konstante Ausdrücke (keine Variablen, außer mapscale)
- Variable Ausdrücke (Variablen aus der Datenquelle, werden mit jedem Datensatz neu ausgewertet – Performance Probleme!)

```
unordered::line::Hauptstraße [Geotyp == "HS"] {
   // Linienbreite ist 10 Meter plus 4 Pixel → Funktion
   line-width: [MeterToPixel(10.0)+4];
   // Text kommt aus der Datenquelle (Spalte: LABEL)
   text: [LABEL];
   // Textfarbe ist Schwarz wenn PARAM1 gleich 3 sonst Blau
   text-color: [PARAM1 == 3 ? "black" : "blue"];
}
```



Beschriftung

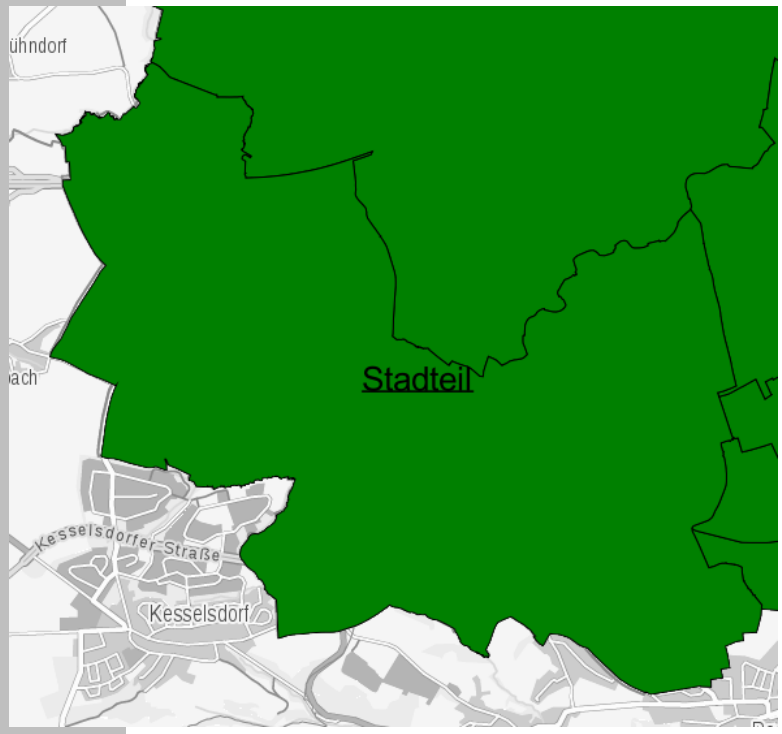
Beschriftung

- Umwandlung von double oder int in string nicht mehr notwendig
- Für jeden Geometrietyp spezielle Parameter um die Beschriftung zu positionieren (Polygon: TODO)
- https://www.cardogis.com/Default.aspx?pgId=1844

- Bei Polygonen (Grundeinstellungen/Rand)
- Bei Linien (… Position, Versatz, text-repeat-distance oder text-orientation (horizontal, alongline, parallel)
- Bei Punkten (…Position, Versatz, Ausrichtung,…) Umsetzung über complex-graphics

Beschriftung Polygon

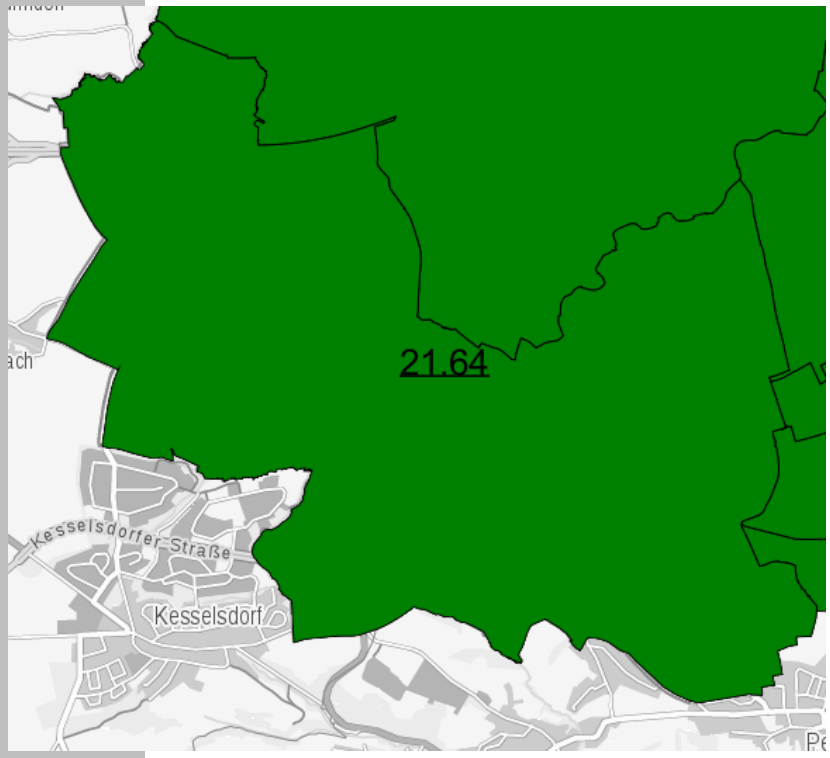
```
polygon{
    //Polygoneigenschaften
    fill-color: green;
    fill-pattern: solid;
    border-line: {
                line-width: 1px;
                line-color: black;
     };
    //Texteigenschaften
    text-color:black;
    text-font-name: "Arial";
    text-style:underline;
    text: "Stadteil";
    text-height:20px;
}
```



Angegebener String wird als Beschriftung gezeichnet.

Beschriftung Polygon

```
polygon{
    //Polygoneigenschaften
    fill-color: green;
    fill-pattern: solid;
    border-line: {
                line-width: 1px;
                line-color: black;
     };
    //Texteigenschaften
    text-color:black;
    text-font-name: "Arial";
    text-style:underline;
    text: [Prozent];
    text-height:20px;
}
```



Wert der Spalte „Prozent“ wird gezeichnet.

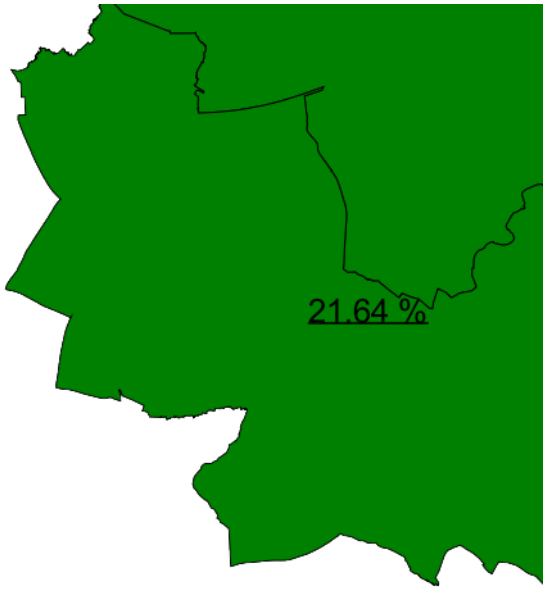
Wiederholung Operatoren

https://www.cardogis.com/Default.aspx?pgId=1839

Logische Operatoren	
&&	logisches und (a && b) -> a und b müssen erfüllt sein
||	logisches oder (a || b) -> a oder b müssen erfüllt sein
String Operationen	
//	Zeichenketten-Verknüpfungsoperator (a // b) -> Beispiel: [ABC // DEF] ergibt ABCDEF
Sonstige Operatoren	
?:	if (wenn) then (dann) else (sonst) Operator (a > b ? 10 : 20) -> das Ergebnis ist 10 wenn a größer als b sonst 20

Beschriftung Polygon

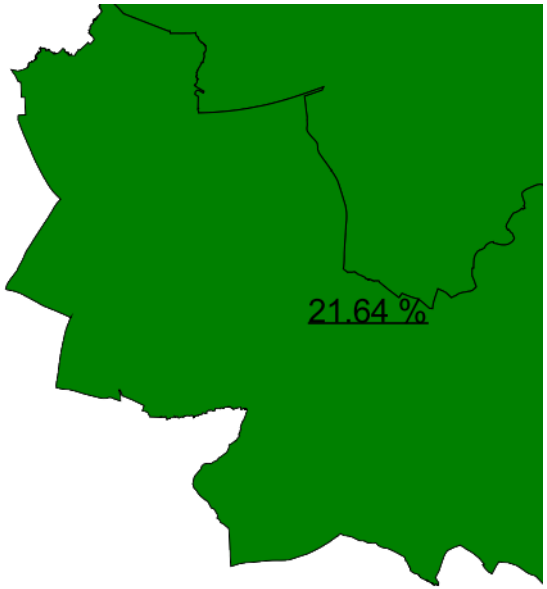
```
polygon{
    //Polygoneigenschaften
    fill-color: green;
    fill-pattern: solid;
    border-line: {
                line-width: 1px;
                line-color: black;
     };
    //Texteigenschaften
    text-color:black;
    text-font-name: "Arial";
    text-style:underline;
    text: [(Prozent) // " %"];
    text-height:20px;
}
```



Wert der Spalte „Prozent“ mit % wird gezeichnet.

Beschriftung Polygon

```
polygon{
    //Polygoneigenschaften
    fill-color: green;
    fill-pattern: solid;
    border-line: {
                line-width: 1px;
                line-color: black;
    };
    //Texteigenschaften
    text-color:black;
    text-font-name: "Arial";
    text-style:underline;
    //If-then-else
    [IsNull(Prozent)?"":Prozent // " % " ];
    text-height:20px;
}
```

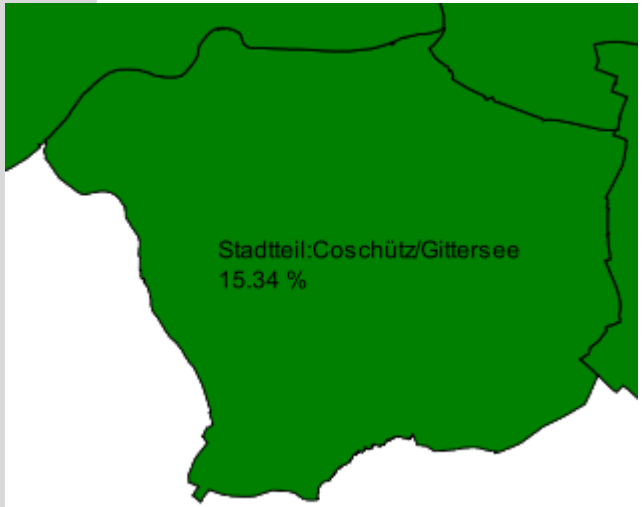


Ist der Wert der Spalte „Prozent“ leer, wird nichts gezeichnet, sonst der Wert der Spalte mit %.

```
//If-then-else
[IsNull(txt)?"":txt];
```

Beschriftung Polygon

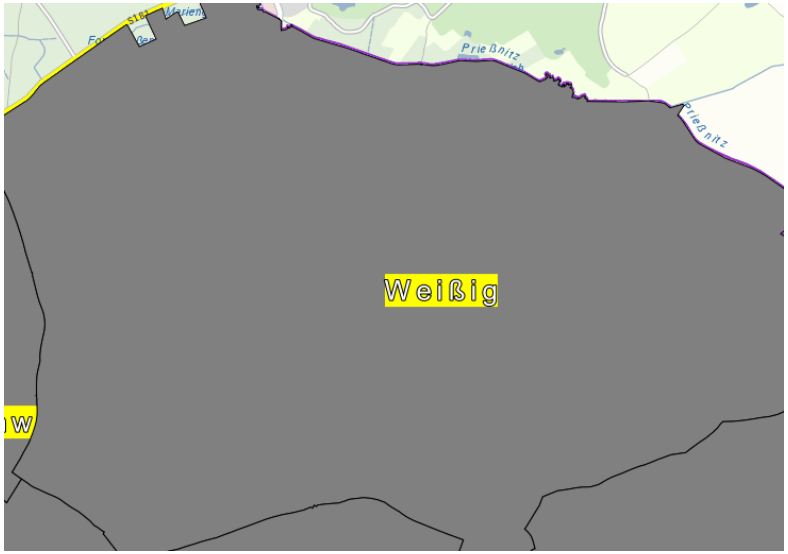
```
polygon{
    //Polygoneigenschaften
    fill-color: green;
    fill-pattern: solid;
    border-line: {
            line-width: 1px;
            line-color: black;
     };
    //Texteigenschaften
    text-color:black;
    text-font-name: "Arial";
    text: ["Stadtteil:" // (stadtteilname) // " @" //
            (Prozent) // " %"];
    text-line-break-character: @;
    text-height:10px;
}
```



Verknüpfung von zwei Spalteninhalten, Prüfung auf NULL kann bei unbekannten Daten sinnvoll sein. Umbruch vorhanden.

Beschriftung Polygon

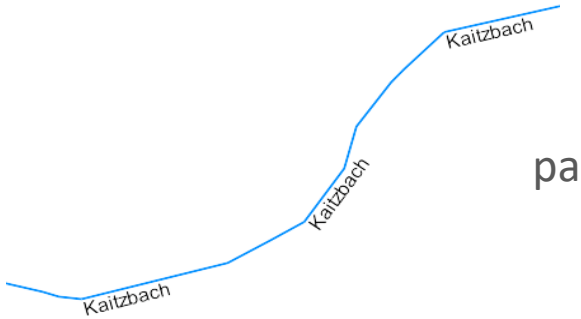
```
polygon{
    //Polygoneigenschaften
...
//Texteigenschaften
    //Texteigenschaften
    text-color:white;
    text-font-name: "Arial";
    text: [(stadtteilname) ];
    text-height:20px;
    text-background-color: yellow;
    text-halo-color: black;
    text-halo-width: 2px;
    text-letter-spacing: 5px;
}
```



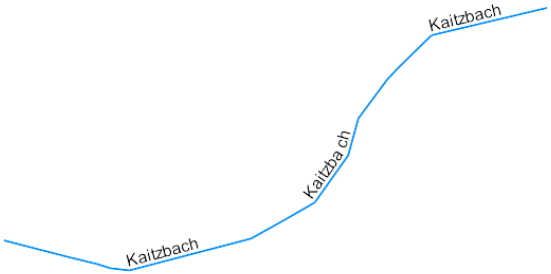
Weitere Gestaltungmöglichkeiten für Beschriftungen.

Beschriftung Linien

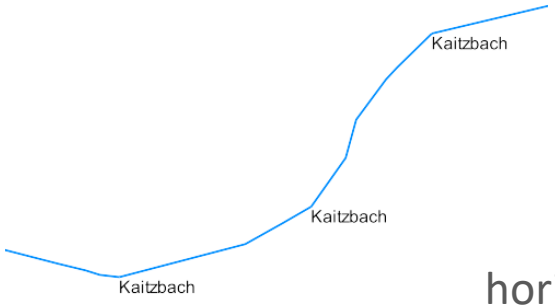
```
//Text-orientation
line {
    line-width: 2px;
    line-color: RGB(0, 143, 255);
    text: [GN];
    text-height: 12px;
    text-color: black;
    text-font-name: "Arial";
    //parallel, alongline, horizontal
    text-orientation: alongline;
}
```



parallel



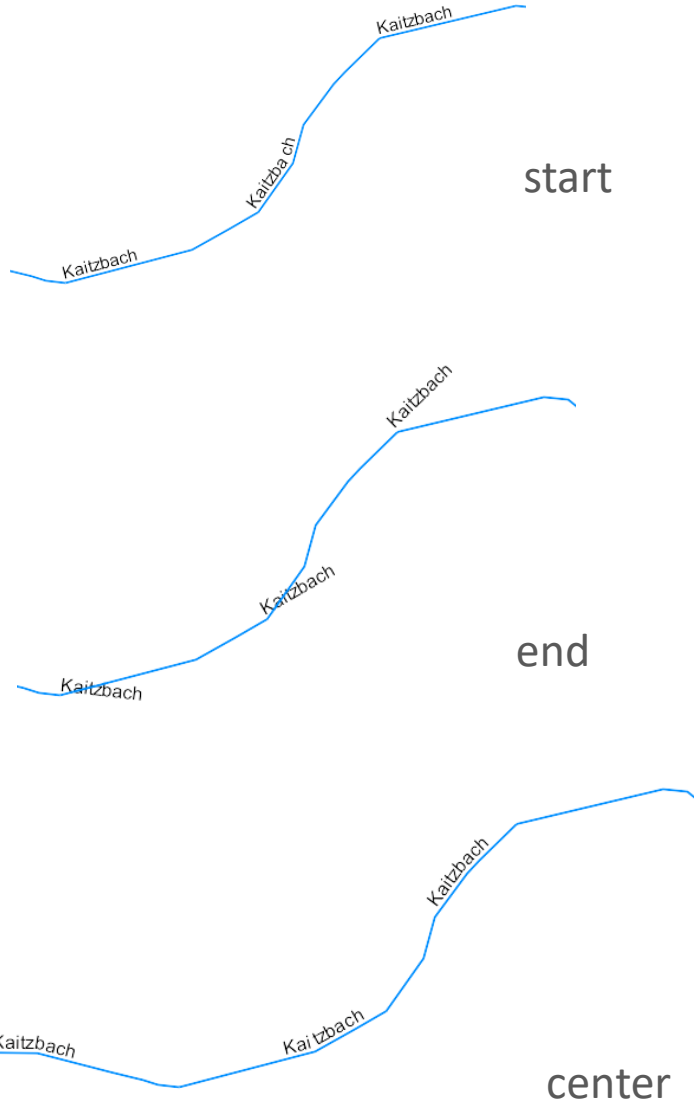
alongline



horizontal

Beschriftung Linien

```
//Text-start-location
line {
    line-width: 2px;
    line-color: RGB(0, 143, 255);
    text: [GN];
    text-height: 12px;
    text-color: black;
    text-font-name: "Arial";
    //parallel, alongline, horizontal
    text-orientation: alongline;
    //start, end, center
    text-start-location: center;
}
```



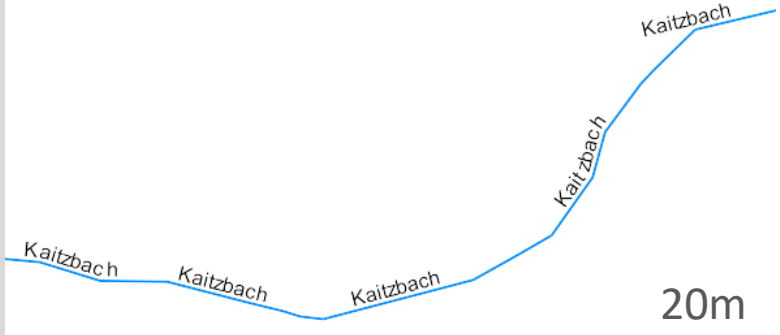
Beschriftung Linien

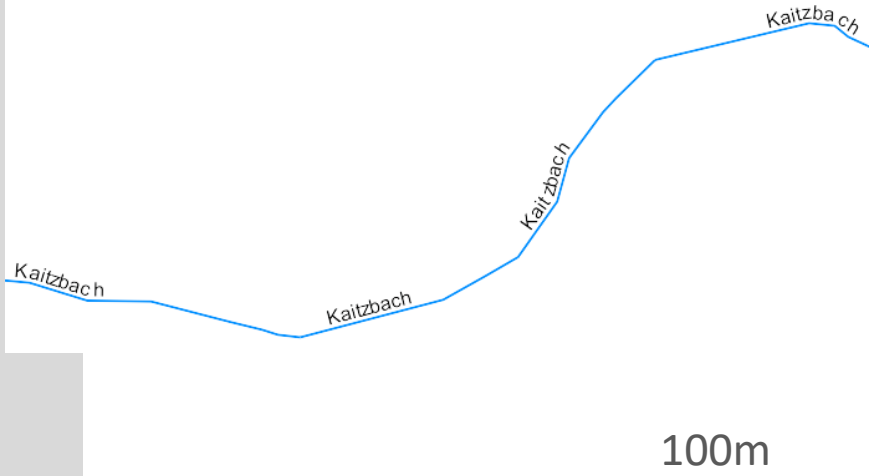
```
//Text-horizontal-alignment
line {
    line-width: 2px;
    line-color: RGB(0, 143, 255);
    text: [GN];
    text-height: 12px;
    text-color: black;
    text-font-name: "Arial";
    //parallel, alongline, horizontal
    text-orientation: alongline;
    //start, end, center
    text-start-location: center;
    //left, right, center
    text-horizontal-alignment: center;
}
```


```
//parallel, alongline, horizontal
    text-orientation: horizontal;
//top, bottom, center
    text-vertical-alignment:top;
```

Beschriftung Linien

```
//Text-repeat-distance
line {
    …
    //parallel, alongline, horizontal
    text-orientation: alongline;
    //start, end, center
    text-start-location: start;
    //left, right, center
    text-horizontal-alignment: left;
     text-repeat-distance: 20m;
// text-repeat-distance: 100m;
}
```

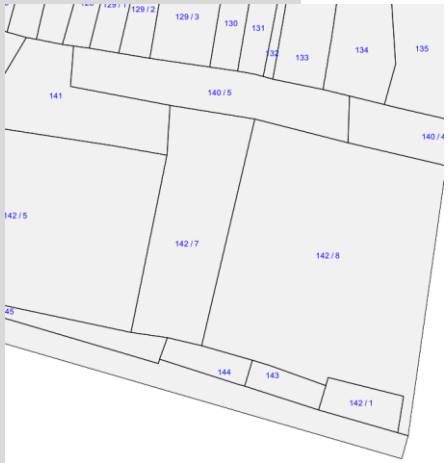




Übung 8

- Flurstücksbeschriftung mit Maßstab

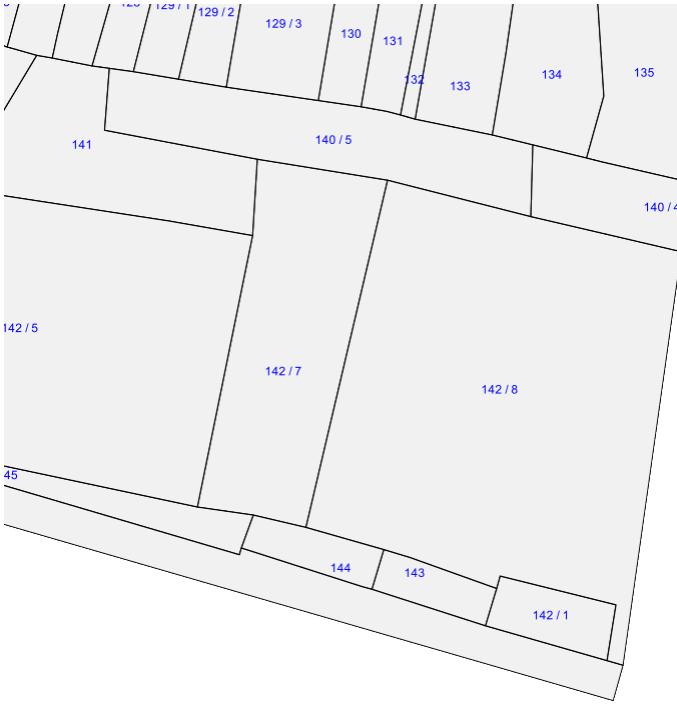
```
unordered {
 polygon {
   fill-color: RGB(128, 128, 128);
   fill-color-opacity: 0.33;
   fill-pattern: solid;
   border-line: {
      line-width: 1px;
      line-color: black;
   };
   text: [(zaehler) // " / " // (nenner) ];
   text-font-name: "Arial";
   text-color: blue;
   text-height: 10px;
 }
}
```




```
ordered {
 polygon {
   fill-color: RGB(128, 128, 128);
   fill-color-opacity: 0.33;
   fill-pattern: solid;
   border-line: {
      line-width: 1px;
      line-color: black;
   };
 }
 polygon [mapscale <= 2000] {
    text: [(zaehler) // " / " // (nenner) ];
    text-font-name: "Arial";
    text-color: blue;
    text-height: 10px;
 }
}
```

Übung 8a

- Flurstücksbeschriftung mit If-then-else Anweisung
- (fiktives Beispiel)
- Ziel: ist der Zähler leer, dann soll nur der Nenner notiert werden, ist der Nenner leer nur der Zähler, sonst beide
- IsNull(zaehler) ? (nenner) : SONST -> erneut Prüfung ob Nenner leer ist
- **text: [IsNull(zaehler) ? (nenner) : IsNull(nenner) ? (zaehler) : (zaehler) // " / " // (nenner)];**



CSS Punkt

```
point {
  complex-graphics:
   circle {
     line-width: 2px;
     line-color: red;
     fill-color: green;
     fill-color-opacity: 0.8;
     fill-pattern: solid;
     radius: 10px;
  };
}
```

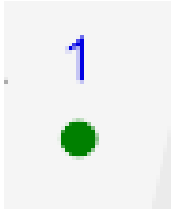
- Punkt wird als komplexe Graphik definiert
- Punkt ist Kreis mit Füllfarbe
- Größe des Punktes über Radius

```
point {
  complex-graphics:
   circle {
              fill-color: green;
              fill-pattern: solid;
              radius: 10px;
  };
}
```

Beschriftung Punkte

- als Text-Block in complex-graphics

```
point {
 complex-graphics:
  circle      { …  },
  text {
              text: [LFDNR];
              text-font-name: "Arial";
              text-color: RGBA(0, 0, 227, 1);
              text-height: 20px;
              text-quality: antialiased;
              text-font-type: symbols;
              //left, right, center
              text-horizontal-alignment: center;
              //top, bottom, center
              text-vertical-alignment: center;
             //Verschiebung des Textes in px oder m möglich
              position-x:0px;
              position-y:20px;
  };
}
```

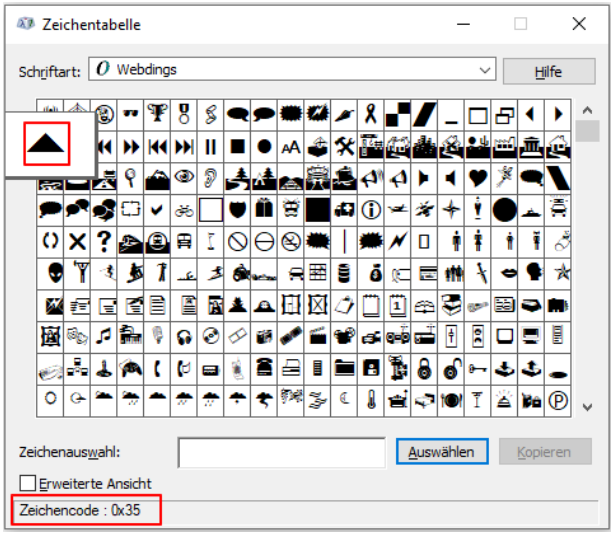


Komplexe Graphiken + Muster

- Kommagetrennte Liste von Blöcken
- Blöcke müssen einen Blocktyp definieren (arc, circle, line, pixmap, polygon, polyline, text, path)
- Können eine ID haben (diese ID kann nicht in der Legende genutzt werden)
- Letzter Block muss mit ; abschließen

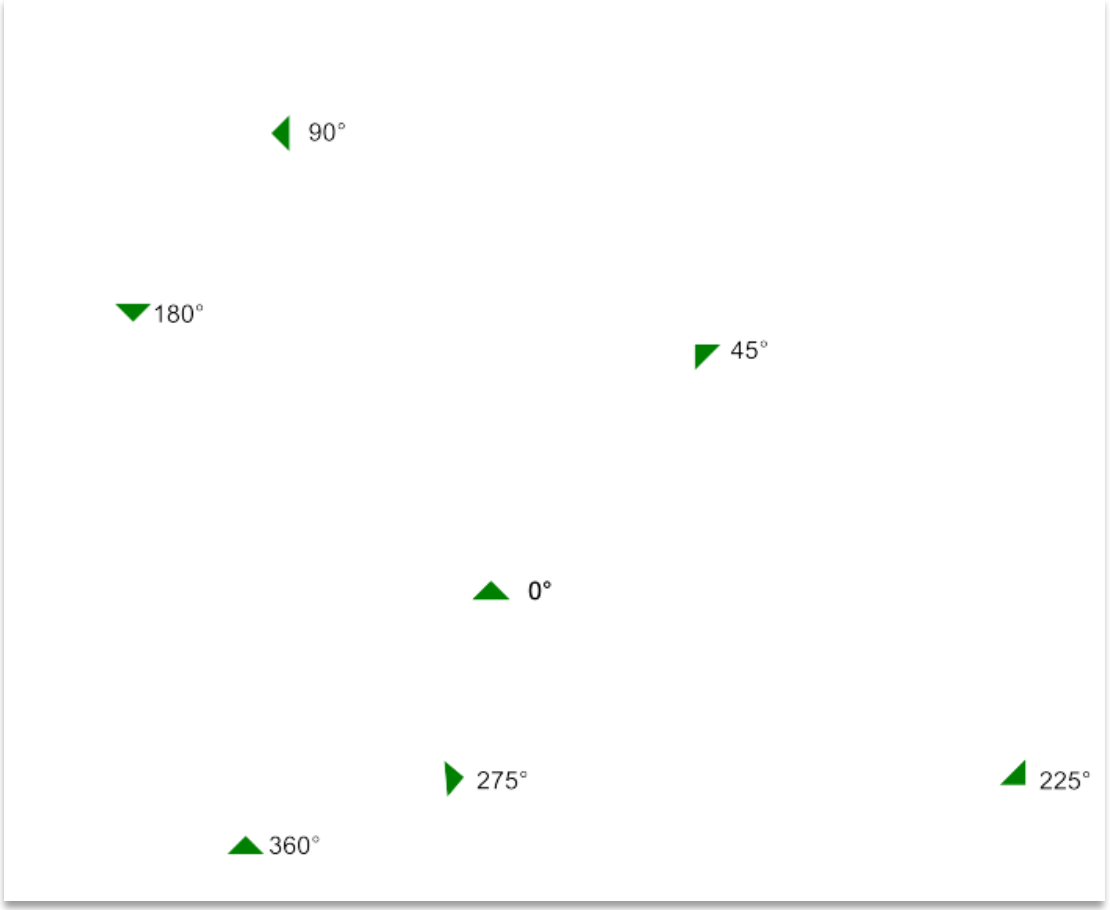
Komplexe Graphiken

```
point {
        complex-graphics:
        text {
                //aus Schriftart Webdings
                //als dezimaler Wert
                text: 53;
                //als hexadezimaler Wert
                text: 0x35;
                //als Tastaturwert
                text: "5";
                text-color: green;
                text-font-name: "Webdings";
                text-height: 40px;
                text-rotation: [winkel];
        };
}
```



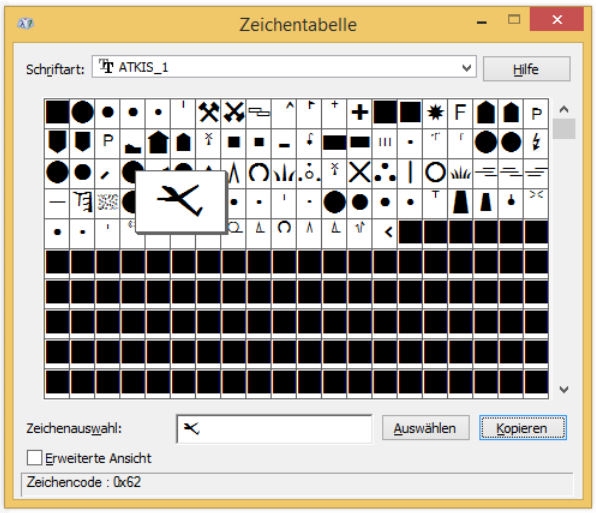
Übung 9 + 9a

- CSS für Punktbeschriftung und Symboldrehung erstellen



Komplexe Graphiken

- Zeichen aus einer Schriftart für Punkte
- Kommagetrennte Liste von Blöcken
- Blöcke müssen einen Blocktyp definieren
- Letzter Block muss mit ; abschließen

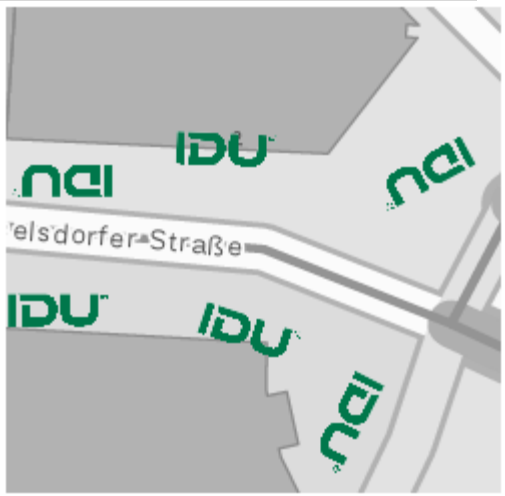


```
single-choice {
 [SIGNATUR === "34000"] {
            point {
            complex-graphics:
            text {
                        text: "`";
                        text-color: RGBA(0, 0, 0, 1);
                        text-font-name: "ATKIS_1";
                        text-height: 100m;
            },
            text {
                        text: "a";
                        text-color: RGBA(255, 255, 255, 1);
                        text-font-name: "ATKIS_1";
                        text-height: 100m;
            },
            text {
                        text: "b";
                        text-color: RGBA(0, 0, 0, 1);
                        text-font-name: "ATKIS_1";
                        text-height: 100m;
            };
}}}
```

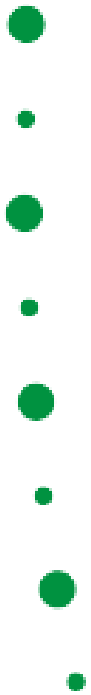
Komplexe Graphiken

- Pixmap

```
point {
  complex-graphics:
    pixmap {
      file-name: ["E:\\cardoSystem\\_Projekt_XY\\Geodaten\\Bilder\\" // typ // ".png"];
      width: 8.0m;
      height: 15.0m;
      opacity: 1.0;
      rotation: [winkel];
      vertical-alignment: center;
      horizontal-alignment: center;
      };
    }
```



Komplexe Muster - Beispiel

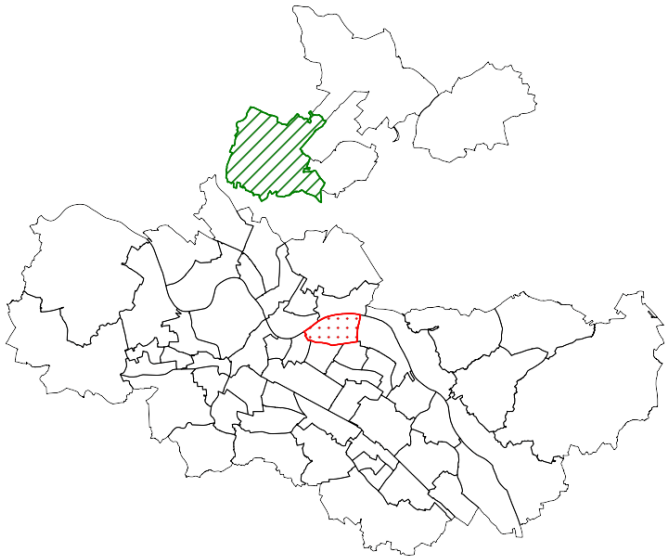


Iwan7CSS für die Darstellung der Ebene:

```
line::id1 [ordn == "2"] {
    linie1 {
        point-placement: interval;
        point-interval: 20m;
        point:  {
            render-quality: antialiased;
            complex-graphics:
            circle::01 {
                fill-pattern: solid;
                radius: 2m;
                fill-color: blue;
            };
        };
    }
    linie2 {
        point-placement: interval;
        point-interval: 40m;
        point: {
            render-quality: antialiased;
            complex-graphics:
            circle::02 {
                fill-pattern: solid;
                radius: 4m;
                fill-color: RGBA(0, 146, 64, 1);
            };
        };
    }
}
```

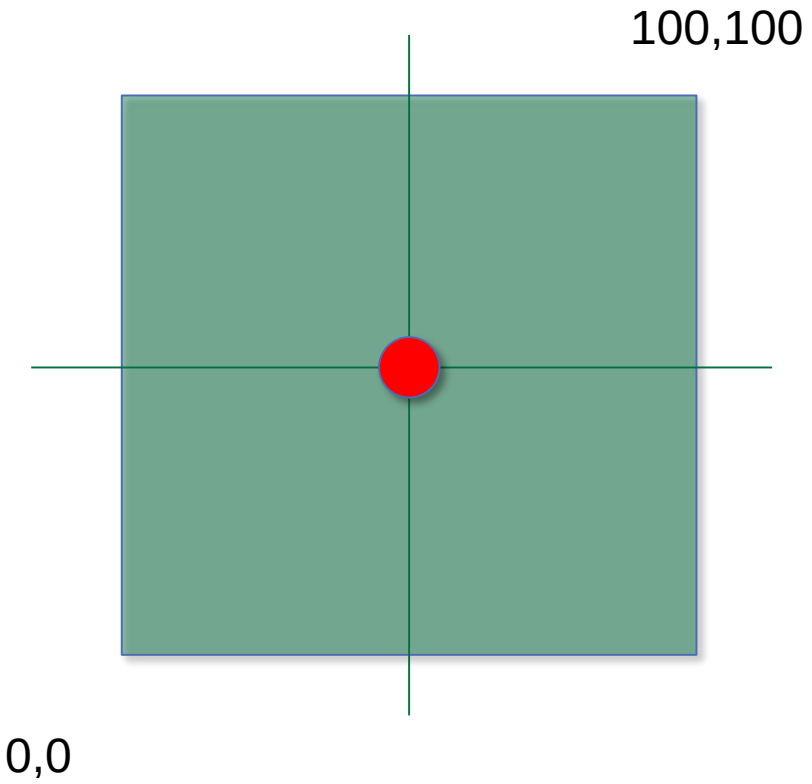
Komplexe Muster

- Komplexe Muster für Polygone, bspw. gepunktete Flächen
- **Eigenschaft: fill-pattern: complex; complex-fill-pattern: …**
- Bbox: Die Breite des komplexen Füllmusters in Metern; dieser Wert hat keinen Einfluß auf die tatsächliche Größe des komplexen Füllmusters sondern dient der Interpretation der Meterangaben der Geometrieattribute



```
polygon::gepunktet [Stadtteilname == "Johannstadt-Nord"] {
    fill-pattern: complex;
    /*Breite des komplexen Füllmusters in m*/
    complex-fill-pattern-bbox-width: 100;
    complex-fill-pattern-bbox-height: 100;
    /*Breite des komplexen Füllmusters in der Karte*/
    complex-fill-pattern-width: 100m;
    /*Mindestbreite des Musters in der Karte in Pixel*/
    complex-fill-pattern-min-width: 10;
    complex-fill-pattern-max-width: 30;
    /*Höhe des komplexen Füllmusters in der Karte*/
    complex-fill-pattern-height: 100m;
    /*Mindesthöhe des Musters in der Karte in Pixel*/
    complex-fill-pattern-min-height: 10;
    complex-fill-pattern-max-height: 30;
    complex-fill-pattern:
        circle::01 {
            line-width: 2px;
            line-color: red;
            position-x: 50m;
            position-y: 50m;
            radius: 10m;
            fill-pattern: solid;
            fill-color: red;
        };
        border-line: {
            line-width: 2px;
            line-color: red;
            line-join: round;
            };
}
```

Komplexe Muster



complex-fill-pattern-bbox-width

complex-fill-pattern-bbox-height

- Virtuelle Box
- Am besten quadratisch

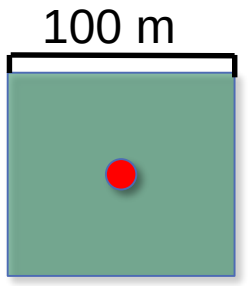
complex-fill-pattern-height

complex-fill-pattern-width

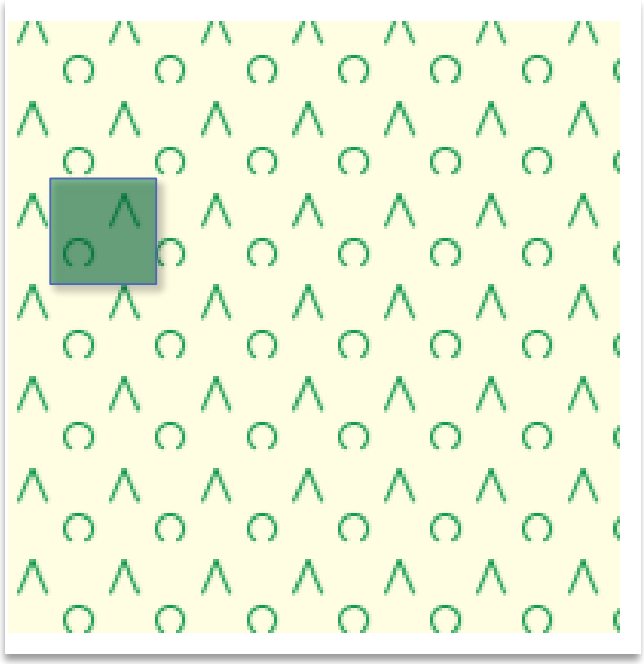
- Tatsächliche Ausgabegröße im m oder px

-> Verhältnis virtuelle Box reale Ausgabegröße muss gleich sein

Komplexe Muster



Komplexe Muster

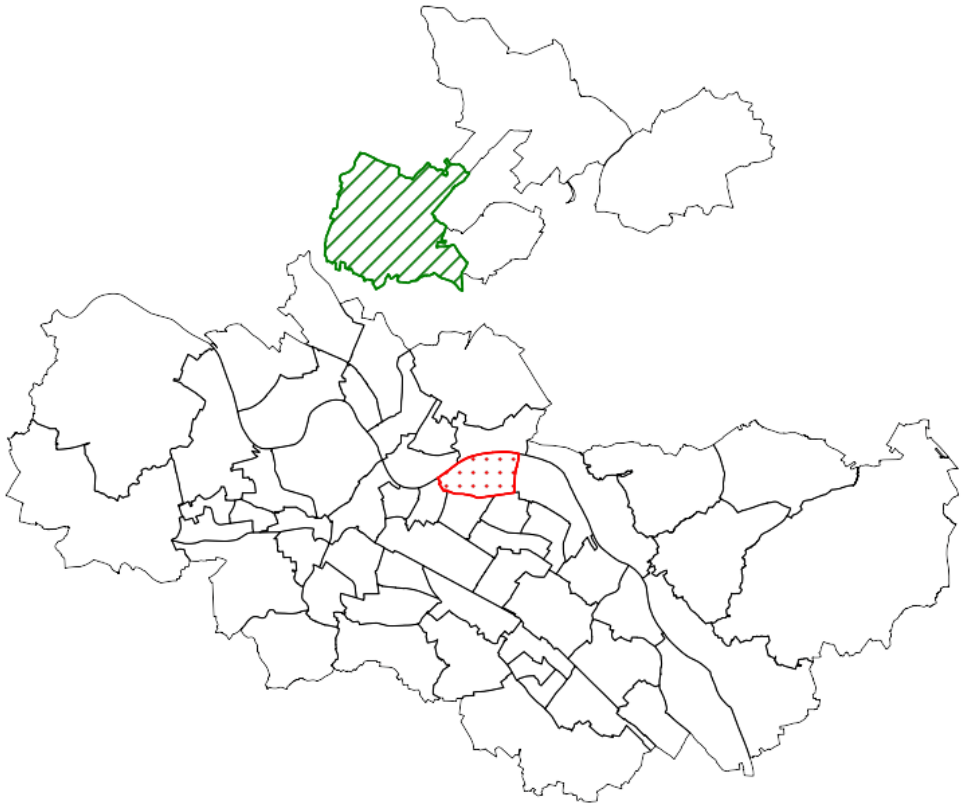


```
polygon {
    Baumschule [Signatur  == "41500_41300_41210"]{
    background-color: RGB(255,254,227);
    complex-fill-pattern-bbox-width: 2000.0;
    complex-fill-pattern-bbox-height: 2000.0;
    complex-fill-pattern-width: 100m;
    complex-fill-pattern-min-width: 20.0;
    complex-fill-pattern-max-width: 30.0;
    complex-fill-pattern-height: 100m;
    complex-fill-pattern-min-height: 20.0;
    complex-fill-pattern-max-height: 30.0;
    complex-fill-pattern:
        arc::01 {
            line-width: 1px;
            line-color: RGB(0, 146, 64);
            position-x: 500.0m;
            position-y: 1500.0m;
            radius: 300.0m;
            start-angle: 120;
            sweep-angle: 300;
        },
        line::01_l {
            line-width: 1px;
            line-color: RGB(0, 146, 64);
            line-cap: round;
            x1: 1200m;
            y1: 100m;
            x2: 1500m;
            y2: 800m;
        },
        line::02_l {
            line-width: 1px;
            line-color: RGB(0, 146, 64);
            line-cap: round;
            x1: 1500m;
            y1: 800m;
            x2: 1800m;
            y2: 100m;
        };

    }
}
```

Übung 10

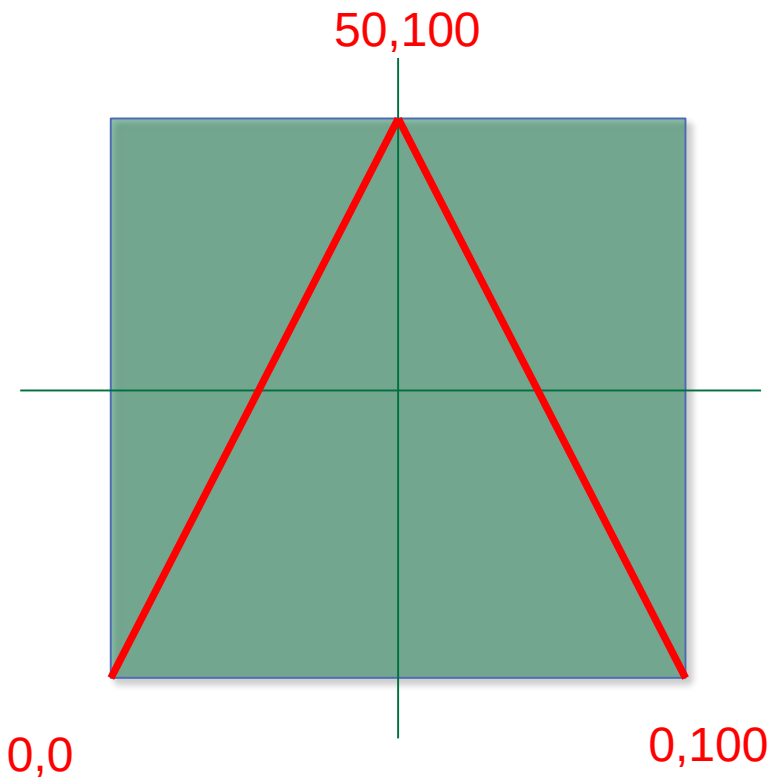
- Gestreifte und gepunktete Fläche erstellen



- BoundingBox definieren
- Punkt als complex-fill-pattern erstellen

Übung 10a

- Gezackte Schraffur erstellen



- BoundingBox definieren
- Gezackte Schraffur erstellen, 2 Linien werden in Virtuelle Bbox gezeichnet

Path Angaben

- Können als Pattern eingefügt werden
- Benötigt path-commands: path-commands: M 24 20 H 36 M 4 20 A 4.600000 4.600000 0 0 0 12 20 M 12 20 A 4.550000 4.550000 0 0 1 20 20;
- SVG Pfade müssen selber erstellt werden
- Beschreibung: https://www.w3schools.com/graphics/svg_path.asp https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Paths
- Programme: Inkscape

Rasterdateien

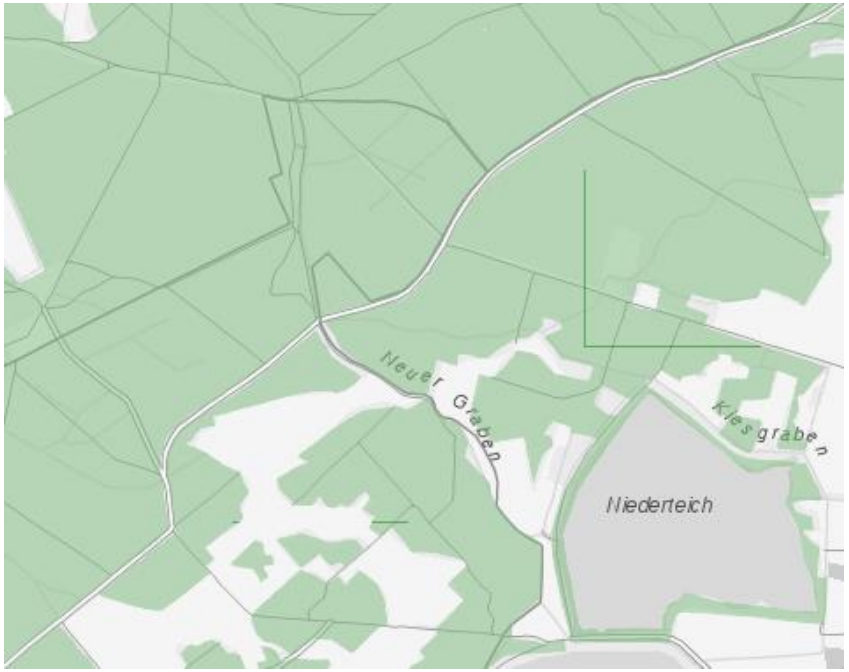
- Georeferenzierte, dateibasierte Rasterdaten
- https://www.cardogis.com/Default.aspx?pgId=1260
- Keine Legende
- Special-raster-properties {…}
  - Angabe von Farbkanälen
  - Bildtransparenz
  - Ersetzen von Farben
  - Transparenz von einzelnen Farben
  - Render-Qualität
  - Graufarben

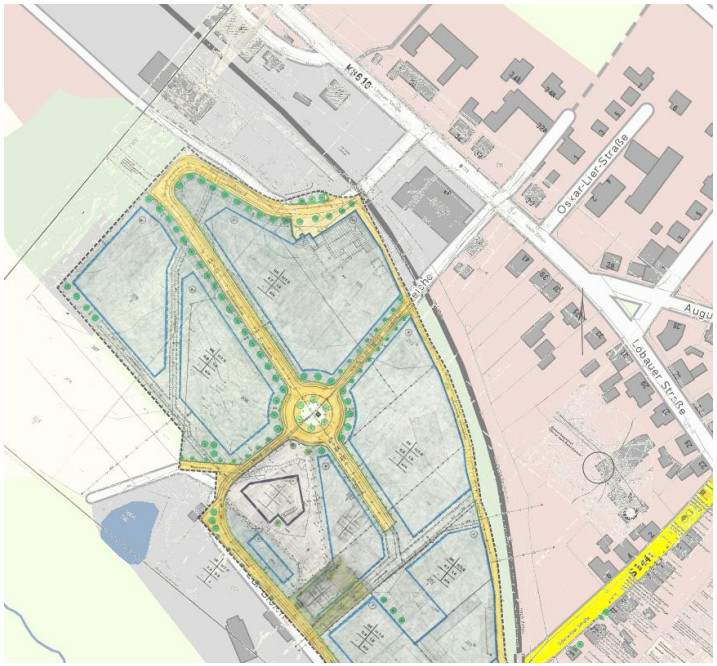
```
special-raster-properties
{
   one-bit-color: RGB(255, 0, 0);
}
```

```
special-raster-properties
{
render-quality:high;
image-opacity: 0.8;
red-channel-index: 0;
green-channel-index: 1;
blue-channel-index: 2;
transparent-colors-and-range: (white, 20.0);
}
```

Übung 11 + 11a

- CSS für Rasterfiles erstellen





Griddateien

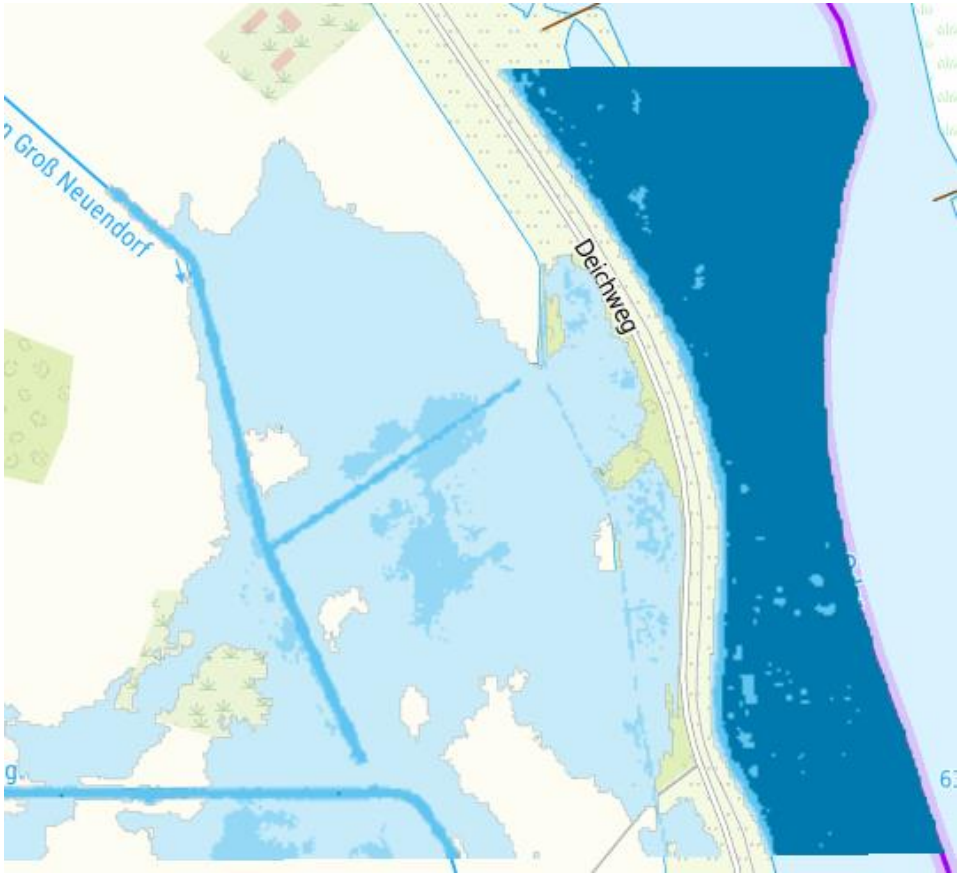
- Betrifft speziell die NetCDF Dateien
- Definition von Farben und Farbverläufen
- Rasterdaten können als GRID-Layer interpretiert werden und damit ist die Nutzung des GRID CSS möglich)
- https://www.cardogis.com/Default.aspx?pgId=1343
- TODO IDU: Legende, wird im Moment automatisch generiert


```
cell-color {
            value: 1;
           color: #C8EBFA;
}
cell-color {
            value: 2;
           color: #94D8F6;
}
cell-color {
            value: 3;
           color: #60C5F1;
}
...
}
```

```
cell-color::id1 {
            start-value: 1;
            end-value: 4;
            color: linear-gradient(lightblue 1, blue,
            darkblue 4);
}
```

Übung 12

- CSS mit linearen Farbgradienten erstellen



Alternative Geometrien

- Gleiches CSS wird für unterschiedliche Geometrien genutzt
- Voraussetzung: DB -> zweite Tabelle oder veränderte Tabelle mit 2. Geometrie (verallgemeinert oder detaillierter…)

Beispiel Fließgewässer -> neuer View mit vereinfachten Geometrien der 2. Ordnung

```
create or replace view datenbrowser.vw_fliesgewaesser as
select *, ST_Simplify(shapegeometry,20) as geom
FROM
datenbrowser.fliessgewaesser
where ordn = '2'
```

- Wichtig: alternative Abfragen müssen gleiche EPSG wie primäre Quelle haben
- Im CSS verwendete Spalten müssen auch in der alternativen Quelle vorhanden sein

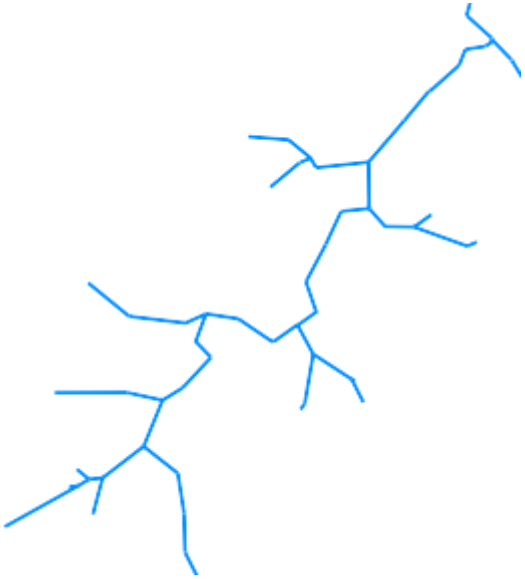
Alternative Geometrien

- Ebene wird geladen
- http://localhost:8287/iwan/project/ui/load
- AlternativeRenderSources angeben
- https://www.cardogis.com/Default.aspx?pgId=1331

Alternative Geometrien

```
[{
        "L249": {
                "type": "PostgresLayer",
                "ConnectionString": "host=milhouse dbname=cardo_anne port=5433 user=cardo
                 password=geheim",
                "Source": "datenbrowser.fliessgewaesser",
                "geomColumnName": "shapegeometry",
                "idColumnName": "cdoautoid",
                "epsgCode": 25833,
                "cssfile": "d:\\...\\uebung1_fliessgewaesser.css",
                "quickLoad": false,
                "onexist": "replaceexisting",
                "alternativeRenderSources": [{
                        "title": "von 10000 bis 50000",
                        "sourceSql": "SELECT * FROM datenbrowser.vw_fliessgewaesser",
                        "geomColName": "geom",
                        "scaleRange": {
                        "minimum": 10000,
                        "maximum": 50000
                        }
                }]
        }
}]
```

Alternative Geometrien



Fragen?

offen

- Positionierung von Schriften im Polygonen
- Legende für Grids
- Gruppierungen von Legenden