FORSCHUNGSGRUPPE
KARTOGRAPHIE
Research Group Cartography


# 127.045 Web Mapping

## Lecture 4: Basic Web Maps

Wangshu Wang, 2019S

# Group Project

- Theme: SDG (Sustainable development goals)
- Your own topics within other themes are welcome as well.
- Post your topic in the TUWEL forum **until May 9th**
- After my confirmation, register your group in the group registration in TUWEL
- Submit an intermediate prototype until June 17th
- Submit the final version to be presented until June 24th, 9:00
- Present your project on June 24th, 12:00-15:00

# Group Project

- Try to be creative!
  - You can also adjust your topic later, as you find out about the details…
  - You can also include JS mapping libraries / techniques we didn't cover in the lecture! (But: no paid-for stuff, no templates / CSS frameworks)

# Organisational Issues

- No lecture next week

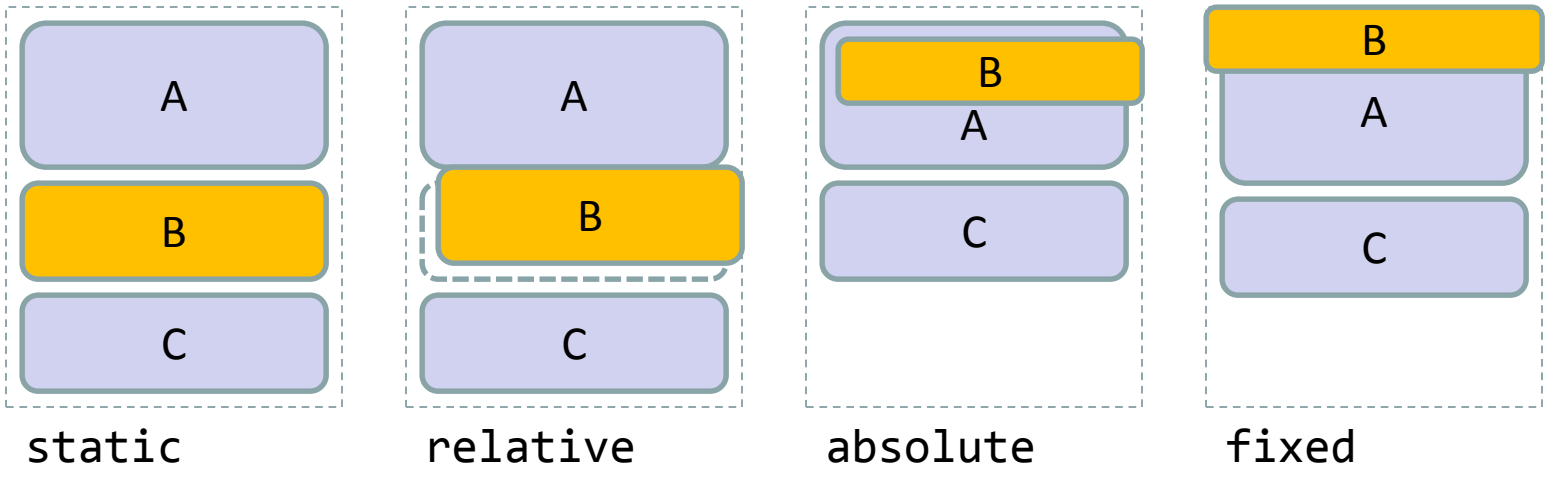
- Final lecture on May 16th:

  Web map design and usability

# CSS POSITIONING

# Positioning

- Position of an element
  - In the flow of content (**`static`**)
  - In the flow, but displaced (**`relative`**)
  - At specific coordinates (**`absolute`**)
  - Fixed to the window/viewport (**`fixed`**)

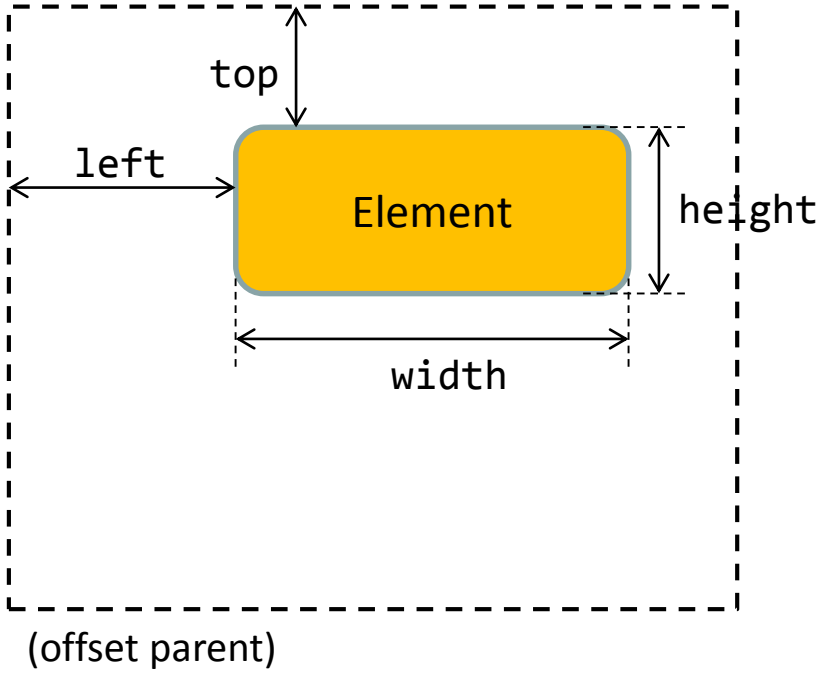


# Positioning

**position: absolute;**

- top, left

```
div.box {
    position: absolute;
    top: 20px;
    left: 30px;
    width: 100px;
    height: 50px;
    background-color: #ffa500;
}
```

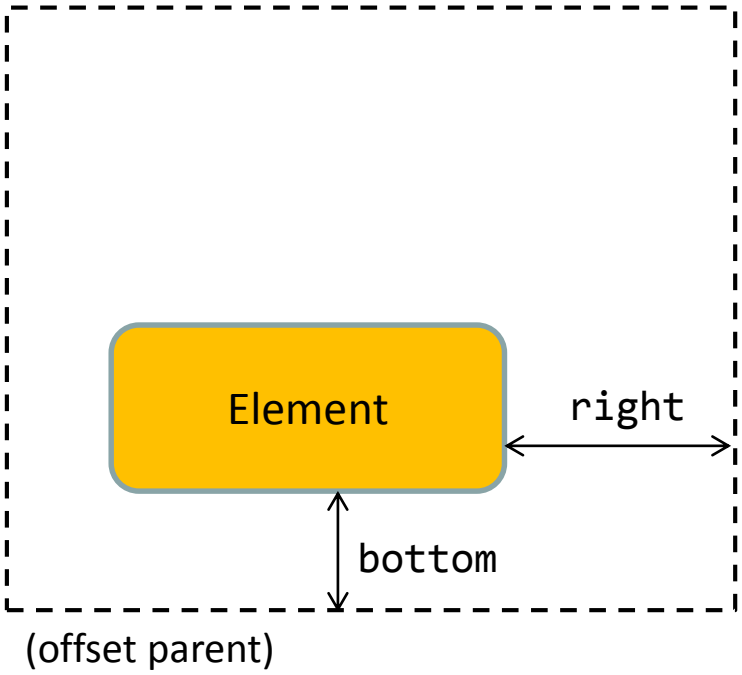


# Positioning

**position: absolute;**

- **bottom, right**

```
div.box {
    position: absolute;
    bottom: 20px;
    right: 30px;
    width: 100px;
    height: 50px;
    background-color: #ffa500;
}
```



# A Primitive “web map”

```
<h1>My first world map!</h1>

<div class="map">
    <div class="marker">
    </div>
</div>
```

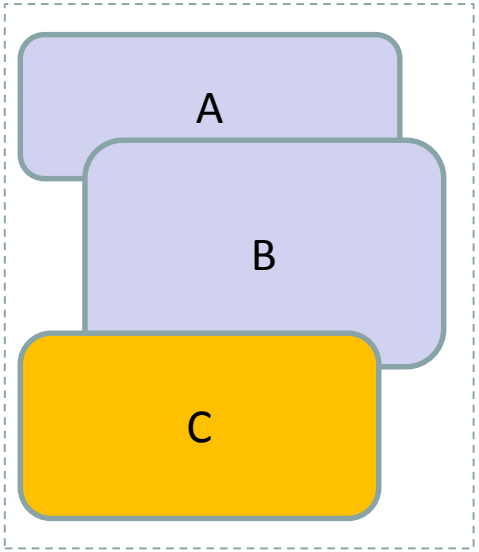
# Positioning

- Offset Parent:
  - Next ancestor with **`position`**
    - **`absolute`**
    - **`relative`** or
    - **`fixed`**

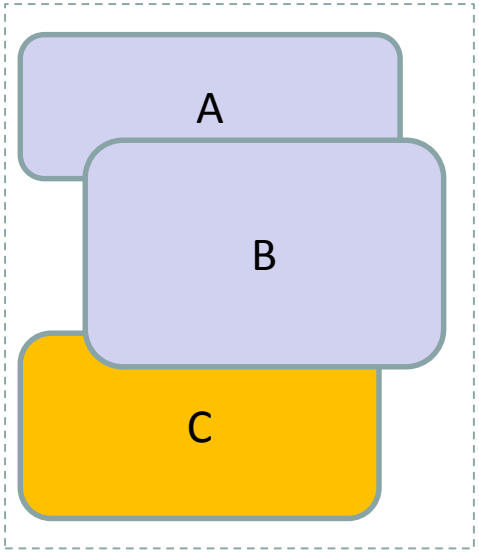
```
.map {
    position: relative;
    /* top = 0, left = 0 (default) */
}

.map .marker {
    position: absolute;
    /* top, left, ... */
}
```

# Layering: z-index

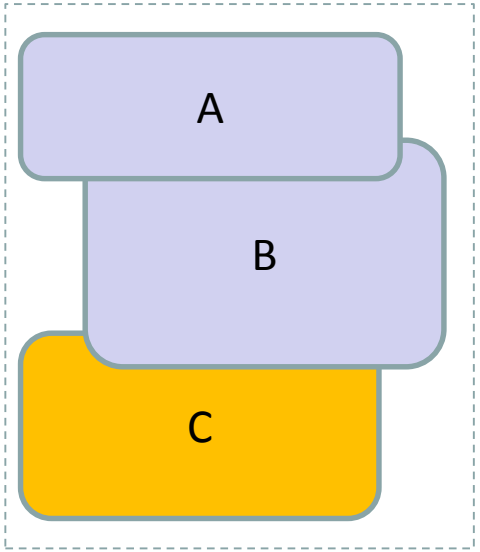
- "Layer order" is determined by **`z-index`** property
  - Higher value is further in front
  - If not specified, later elements are further in front




`default`




```
B { z-index: 1 }
```




```
B { z-index: 1 }
A { z-index: 10 }
```

# A Primitive “web map”

```
/* in .css file or "style" tag in head */
.map {
    position: relative;           /* Make offset parent for markers! */
    width: 512px;
    height: 512px;
    background-image: url('worldmap-mercator.png');
}
.map .marker {
    position: absolute;
    z-index: 1;
    width: 25px;                        /* Size of marker image */
    height: 41px;
    background-image: url('marker.png');
}
```

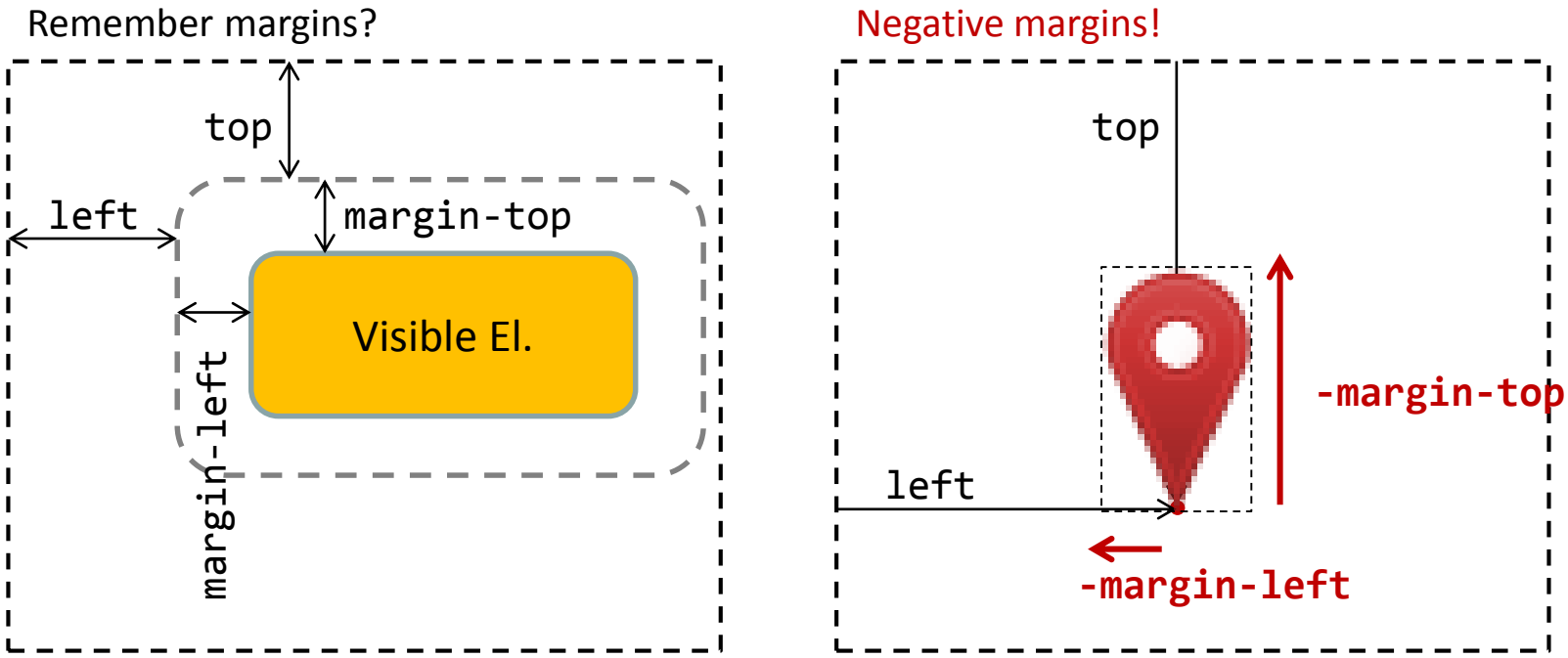
# A Primitive “web map”

```
/* in .css file or "style" tag in head */
.map {
    position: relative;
    width: 512px;
    height: 512px;
    background-image: url('worldmap-mercator.png');
}
.map .marker {
    position: absolute;
    z-index: 1;
    width: 25px;
    height: 41px;
    top: 50%;                          /* point to 0° 0° for now */
    left: 50%;
    background-image: url('marker.png');
}
```

simplemap.html

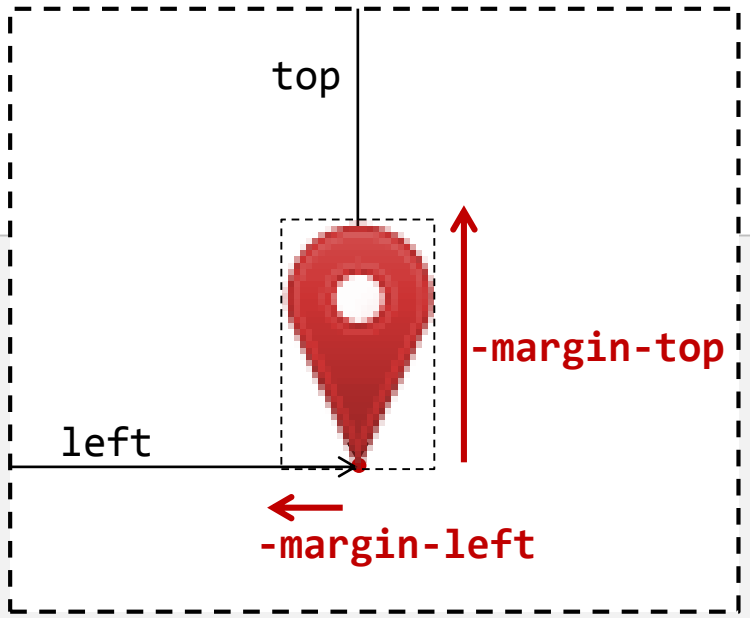
# A Primitive “web map”

## Positioning the marker



# A Primitive “web map”

Positioning the marker




```
.map .marker {
    position: absolute;
    width: 25px;
    height: 41px;
    top: 50%;
    left: 50%;
    margin-left: -12px;
    margin-top: -42px;
    background-image: url('marker.png');
}
```

# A Primitive “web map”

Adding coordinates to the marker

```
<div class="map">
    <div class="marker">
        <!-- lat=48.2 lon=16.4 -->
    </div>
</div>
```

# A Primitive “web map”

Adding map coordinates

```
<div class="map">
    <div class="marker"
         style="top: 34.7%; left: 54.5%;">
        <!-- lat=48.2 lon=16.4 -->
    </div>
</div>
```

# A Primitive “web map”

Making the map scalable

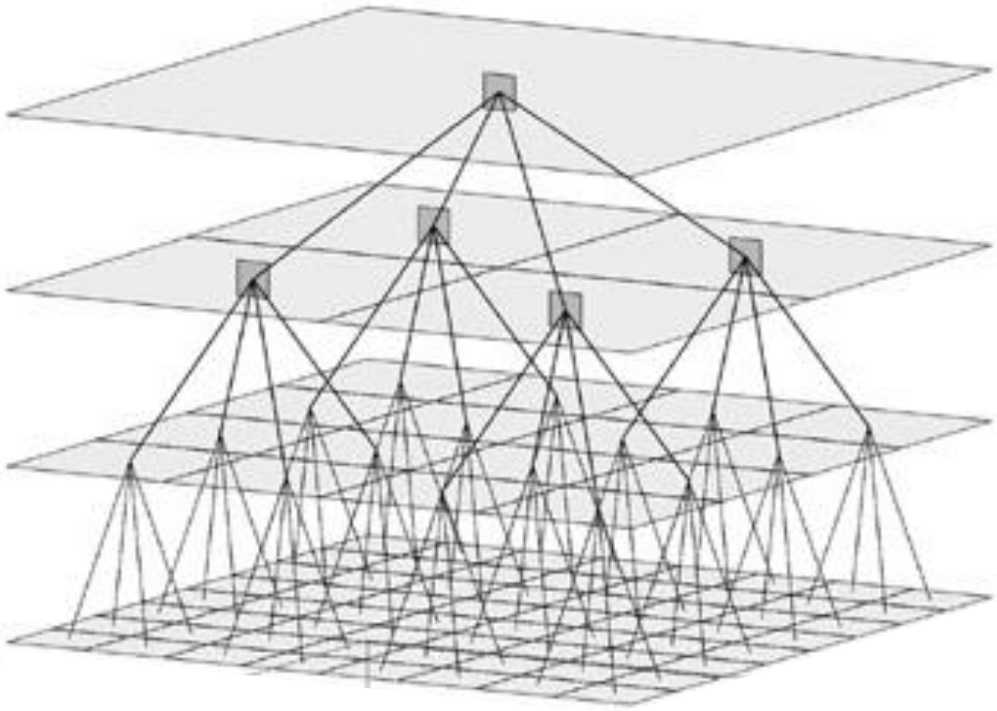
```
.map {
    position: relative;
    width: 512px;
    height: 512px;
    background-image: url('worldmap-mercator.png');
    background-size: 100% 100%;
}
```

simplemap_solution.html

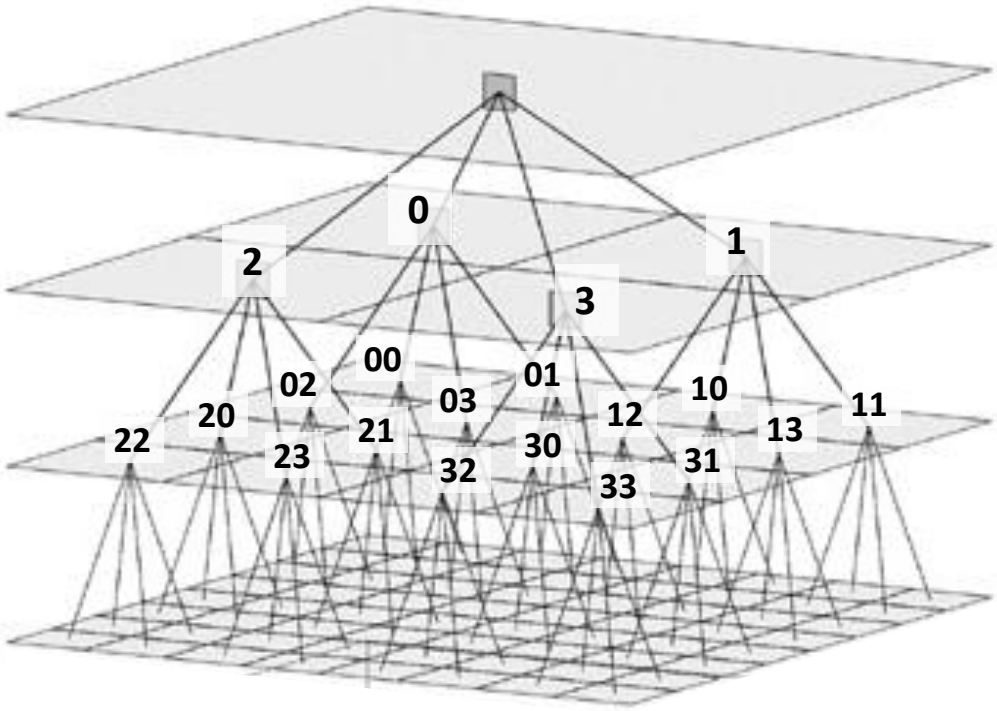
# ZOOMING & PANNING MAPS

# Zooming & Panning

# Quadtree



# Quadtree




Zoom level 0

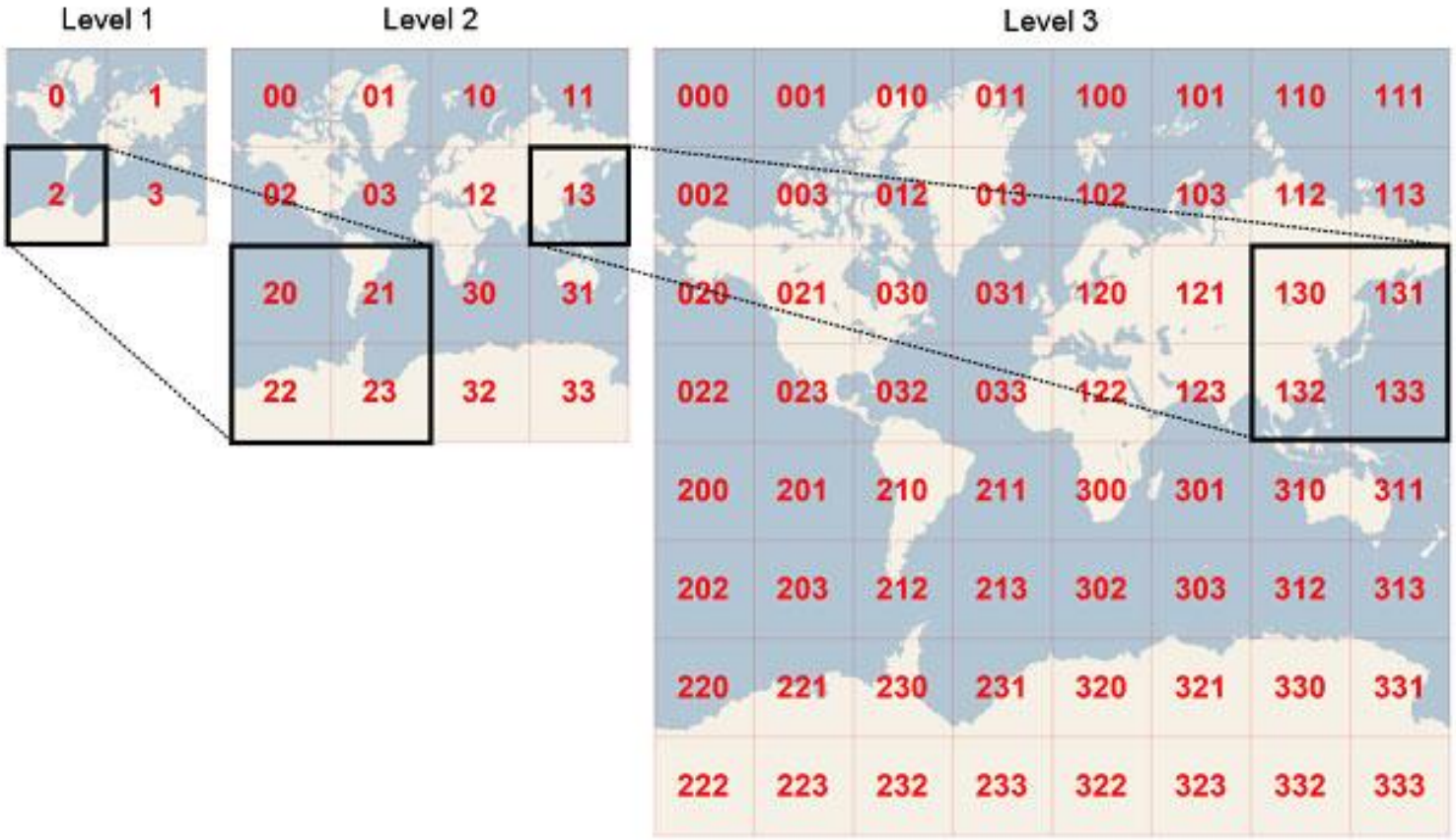
Zoom level 1

Zoom level 2

Zoom level 3

...

# Quadtree




*Source: Microsoft Virtual Earth SDK*

# Quadtree

- On zoom level *n*:
  - $4^n$ tiles
  - Map size: $2^n \times$ tile size
  - Tile address: n letters (n × 2 Bit)

- Example:
  Zoom level 10, tile size 256

  - 1.048.576 tiles
  - 262.144 pixels wide
  - Tilename e.g. `120/230/1132.png`

# Web Mercator Projection

- Simplified Mercator Projection
  - WGS 84 datum (ellipsoidal, GPS)
  - Spherical formulas used for all scales
  - Deviates up to 35 km from true Mercator (!)
  - Cut-off at 85.051129° N/S → square map!

- Formulas for a 256-pixel map:

$$x = \frac{128}{\pi} 2^{\text{zoom level}} (\lambda + \pi) \text{ pixels}$$

$$y = \frac{128}{\pi} 2^{\text{zoom level}} (\pi - \ln \left[ \tan \left( \frac{\pi}{4} + \frac{\varphi}{2} \right) \right]) \text{ pixels}$$

# Web Mercator Projection

“The NGA Geomatics Office has assessed the use of Web Mercator [...] may cause errors up to 40,000 meters. This erroneous geospatial positioning information poses an unacceptable risk to global safety of navigation activities”

(US National Geospatial Intelligence Agency)

# “Slippy” Style Web Maps

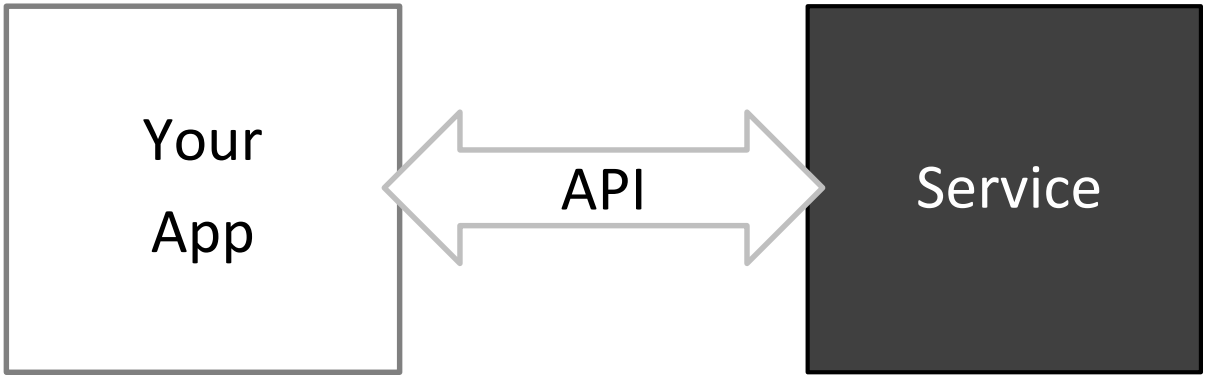
An interactive web map that can zoom + pan:

- Arrange **`img`** / **`div`** squares to cover the map area
- Use “wrapper” **`div`** to cut out desired size
- Figure out the correct quadtree tiles to load (Depending on zoom level + location)
- When user moves mouse with button pressed, shift map (using CSS) and add new tiles if needed
- When user zooms, load new tiles

# WEB MAPPING APIS

# Web Mapping APIs

- API: **A**pplication **P**rogramming **I**nterface
  - Standardized way to make use of external program code
  - Open Source or proprietary (closed source)
  - External code is "black box"



# Web Mapping APIs

Core functions:

- „Slippy Map“ paradigm (pan, zoom)
- Manage tile loading
- Let you add map features (Points, lines etc.)

Additional functions:

- More interaction (Map Controls, Editing)
- Data loading & visualization
- Geocoding (Place Name ↔ Location)
- Different map styles
- …

# Web Mapping APIs We Will Use

## Google Maps JavaScript API

- Released 2005
- Closed-Source
- Backed by Google infrastructure (tile server, geocoder , …)
- You need an API Key to use it!

## Leaflet

- Open Source
- Released 2011
- Current Version: 1.4.0
- No infrastructure provided (but can use any tiles)

# Adding JavaScript to HTML

- Web Mapping APIs come as JavaScript files

- In the **head** element of your HTML page, add

Google Maps

```
<script
src="http://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY">
</script>
```

Leaflet

```
<link rel="stylesheet"
href="http://unpkg.com/leaflet@1.0.3/dist/leaflet.css">
<script
src="http://unpkg.com/leaflet@1.0.3/dist/leaflet.js"></script>
```

# Adding a Map Element

- Inside the **body** of your HTML page, add

```
<div id="map">
</div>
```

- And style it through CSS:

```
#map {
    width: 512px;
    height: 512px;
}
```

# Adding JavaScript to HTML

- Before **</body>** in your HTML page, add

Google Maps

```
<script>

var mapOptions = {
    center: {lat: 48.2, lng: 16.4},
    zoom: 10
};
var mapEl = document.getElementById('map');

var map = new google.maps.Map(mapEl, mapOptions);

// TODO: Add markers, ...

</script>
</body>
```

google-map.html

# Adding JavaScript to HTML

- Before **</body>** in your HTML page, add

Leaflet

```
<script>

var map = L.map('map').setView([48.2, 16.4], 10);

var layer = L.tileLayer(
  'http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'
);
layer.addTo(map);

// TODO: Add markers, ...

</script>
</body>
```

leaflet-map.html

# Adding Markers

Google Maps

```
// Add a marker
var pos = {lat: 48.196857, lng: 16.370766};
var marker = new google.maps.Marker({
    position: pos,
    map: map,
    title: "Lecture Room EI8"
});
```

Leaflet

```
// Add a marker
var pos = [48.196857, 16.370766];
var marker = L.marker(pos, {
    title: "Lecture Room EI8"
});
marker.addTo(map);
```

# Adding Geometry

Google Maps

```
// Add a rectangle

var bounds = new google.maps.LatLngBounds(
    new google.maps.LatLng(48.1, 16.3),
    new google.maps.LatLng(48.3, 16.5)
);
var rectangle = new google.maps.Rectangle({
    strokeColor: '#0000ff',
    strokeWeight: 2,
    fillColor: 'transparent',
    map: map,
    bounds: bounds
});
```

# Adding Geometry

Leaflet

```
// Add a Rectangle

var bounds = [[48.1, 16.3], [48.3, 16.5]];
var rect = L.rectangle(bounds, {
    color: "#0000ff",
    fill: false,
    weight: 2
});
rect.addTo(map);
```

# Changing the Marker Icon

Google Maps

```
// Add a Marker with our own icon

var image = 'home.png';

var pos = { lat: 48.196857, lng: 16.370766 };
var marker = new google.maps.Marker({
    position: pos,
    map: map,
    title: "Lecture Room EI5",
    icon: image
});
```

# Changing the Marker Icon

Leaflet

```
var icon = L.icon({
    iconUrl: 'home.png',
    iconSize: [32, 37],
    iconAnchor: [16, 37]
});
var pos = [48.196857, 16.370766];
var marker = L.marker(pos, {
    title: "Lecture Room EI5",
    icon: icon
});
marker.addTo(map);
```

# Working with Mapping APIs

- API Documentation
  - Google Maps API
    - Developer's Guide
    - API Reference
  - Leaflet
    - Tutorials
    - API Reference

- If you consider using Google Maps in a real-world project, read about Usage Limits and Billing!
- If you consider using Leaflet in a real-world project with lots of visitors, you need to find a different tile server! → Leaflet Tile Providers

# ASSIGNMENT & OUTLOOK

# Assignment 2

- Create 1 page with 2 maps on them
  - 1 manually coded, like in the first part of the lecture
  - 1 using an API (Google Maps or Leaflet)
- Add content:
  - Manually coded map:
    - 3 Markers, 1 Rectangle

      (Hint: use `top, left, width, height, border`)
  - API map:
    - min. 3 Markers, 1 Area (rectangle or other shape)
    - Add InfoWindow (GMaps) / Popup (Leaflet) to each
      - Find out about those by yourself

# Assignment 2

- Required
  - Your own images for marker(s) for both maps
    - Create in Photoshop, Gimp (→ VU Geomedia Techniques)
    - at least one
- Optional
  - Integration with previous assignment
- Bonus points
  - You own map image for manually coded map (any area)
  - More different content (Geometry, Icons etc.)
- Due: May 29th
  - Open Q&A: May 23rd, 14:00-16:00 EI8

# Next Time

- No lecture next week!
- Final lecture on May 16th:

  Web map design and usability
- Preparation in groups before the lecture
- Short presentation (2 minutes per group), no slide
- Will not be graded

# Assignment: Evaluating Web Maps

- Choose a web mapping application of your interest.
- Explore the following general questions:
  - What is the **main purpose** of the mapping application?
  - Who is the **publisher** of the map?
  - Who do you think is the **target audience**?
  - Can the **data sources** be easily identified?
  - Which **technology** is used for the map?
- Explore the following usability questions:
  - Could you easily identify the functionalities?
  - Were there explanations / help files?
  - Did you enjoy using the web map?
  - Is there anything you especially like or dislike about the application?