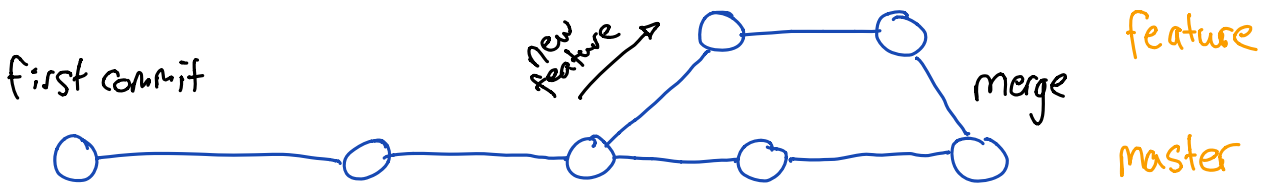
# GIT

- branching let's use have versions of code in parallel




HEAD is position where you are currently in the repo

- git branch — list of branches in repo
- git branch <name> — create new branch
- git checkout <name> — switch to branch / commit
- git merge <name> — master branch needs to be active
  ↳ merges commits of branch into master

- git push --set-upstream origin <name>
  push new branch to remote server

## CONCEPT OF REMOTES

- version of repository that lives somewhere else, e.g. Github
- git fetch — download version from remote
  git merge <origin/master> — update master to version of remote
  ↳ git pull

- Forks : create entirely seperate duplicate of repository
  ↳ used in Open Source Contribution

- Pull Request : Bring in your changes to a branch of another repository or another branch in same repo

## HTML

<a href="file.html"> Text </a> creates a link using anchor tag (href = hyperlink reference)

<a href="#idOfElement"> links to element within same webpage

- Organization :
  - <header>
  - <nav> simple way to organize
  - <section> webpage and style sections
  - <footer>

- HTML5 tags :
  - <audio> embed media within page
  - <video>
  - <datalist> list with values that can be used for autocompletion

- Input types :
  - text
  - password
  - radio give same name, define val
  - list = "<datalist>" datalist with id and option elements with value

# CSS

tag1 tag2 { ... } — styles all tag2 elements that are descendant/contained within tag1
descendant selector

tag1 > tag2 { ... } — selects only tag2 elements that are immediate children of tag1
child selector

tag [ attribute = value ] { ... } — select elements that contain specified attribute
attribute selector

- pseudoclass : specific state of element
  - tag : hover { ... } style when cursor hovers over element
- pseudoelements :
  - tag :: before {
    content : "Text";
    }
    pseudoelement that is added before every tag element (but inside element)
  - tag :: selection { ... } style selection

# Responsive Design

- different screen sizes require different element sizes
- elements might have to move around
- Media Queries → apply style to specific "media"
  ↳ e.g. device of specific size

```
@media print {
    tag { ...}
}
```

only when page is about to be printed

```
@media (min-width: 500px)
```

style for screen widths >500px

```
@media (max-width: 499px)
```

style for smaller screen sizes

```
<meta name="viewport" content="width=device-width,
                                initial-scale=1.0">
```

↳ sets viewport of webpage to device width

- Flexboxs : cells automatically rearrange according to view width

  ↳ div container ⇒ `display: flex; flex-wrap: wrap;`

- Grid: size of specific cells adjust

  ↳ div grid ⇒ `display: grid; grid-column-gap: 20px;`
  `grid-row-gap: 15px;`
  `grid-template-columns: 100px 75px auto;`

# CSS LIBRARIES

- `<link rel="stylesheet" href="url_to_bootstrap.css">`

- Bootstrap uses column-based model (rows with 12 colums each)

```
<div class="container">
    <div class="row>
        <div class="column-3"></div>
    </div>
</div>
```

↳ streches over 3 colums

- `class="col-lg-3 col-sm-6"`

↳ on large screen 3 out of 12 colums

→ on small screen 6 out of 12 colums (half)

- easily create alert messages

- Documentation on https://getbootstrap.com

# VARIABLES IN CSS / SASS

- can be used as constants for sizes, colors etc.
- SASS allows to programmatically create stylesheets → .scss

```
$variableName: value;
css-property: $variableName;
```

- webbrowser don't understand SASS out-of-the-box

`sass file.scss file.css` → compile scss into css file

`sass --watch file.scss:file.css` → automatically recompiles changes

## NESTED CSS SELECTORS WITH SASS

```
div {
    property: value;
    tag1 {
        property: value;
    }
}
```

structure of CSS code reflects which elements are selected

## INHERITANCE WITH SASS

```
%template { ... }

.class {
    @extend %template;
    ....
}
```

creates template with CSS properties

load template

add other properties