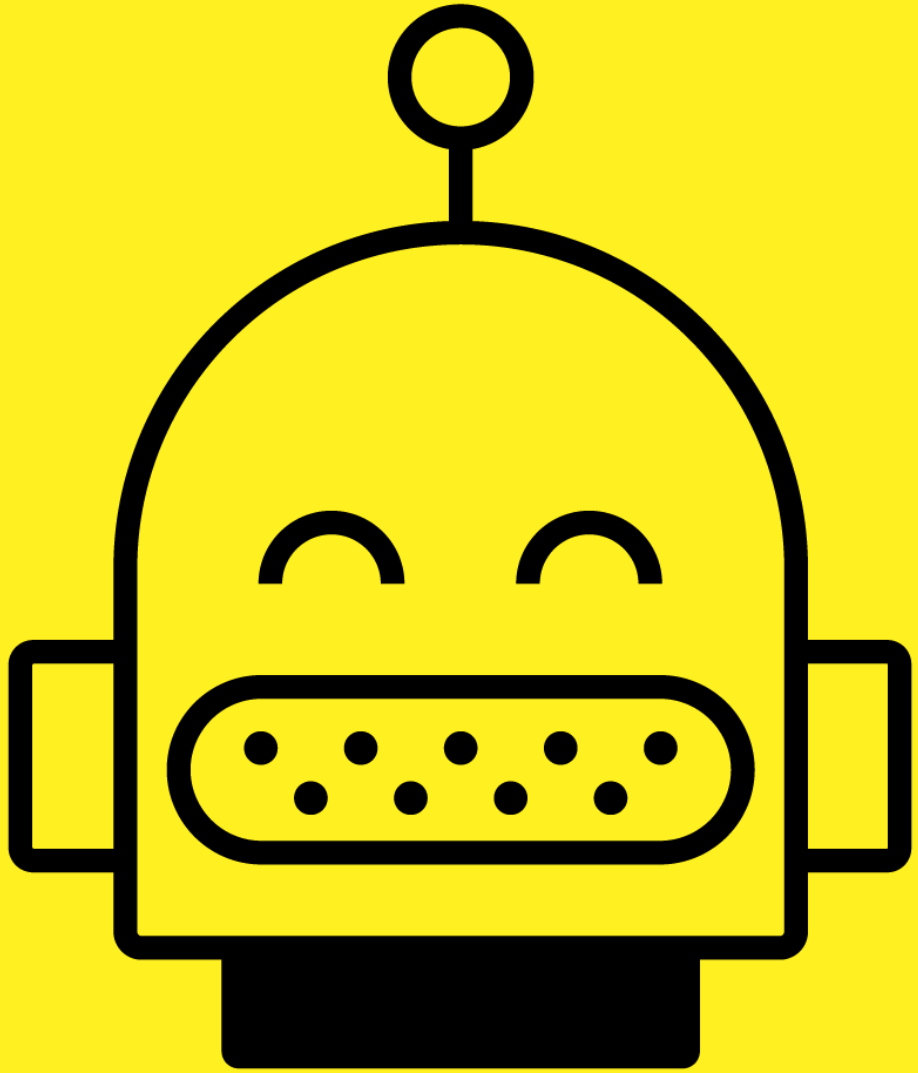
Communication and Multimedia Design | Hogeschool van Amsterdam

Project Tech

Les 4-2

Github voor teams, Frontend optimalisatie



Creating Tomorrow

Planning

(1)

Github voor teams

Branches, pull request, merging

(2)

Frontend optimalisatie

Responsive design, progressive enhancement, coding standards, ES6, linters

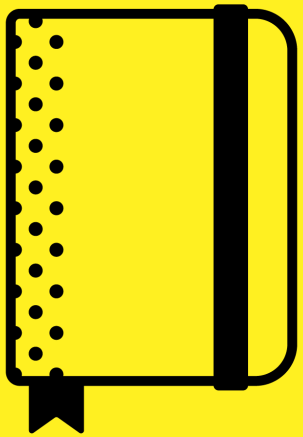
(3)

Pauze

(4)

Lesopdracht

ESLint



Theorie

Git(Hub) voor teams

https://guides.github.com/introduction/flow/

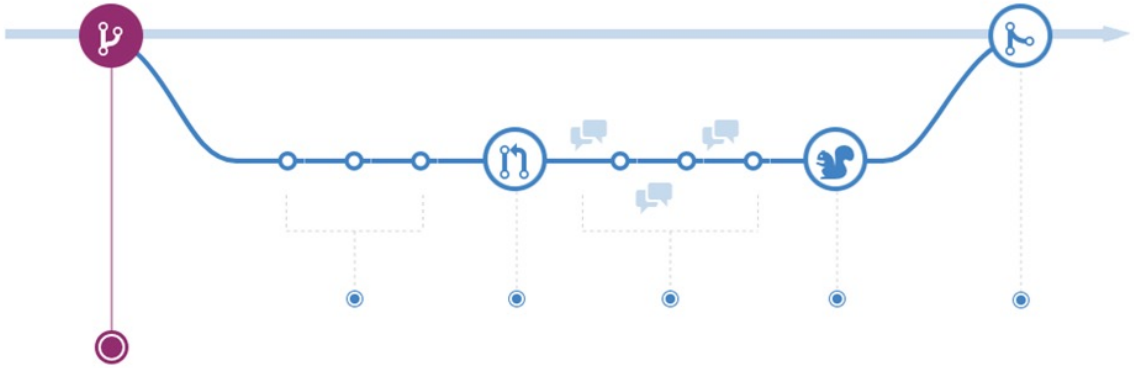
GitHub Guides

Video Guides GitHub Help GitHub.com

Understanding the GitHub flow

5 minute read Download PDF version

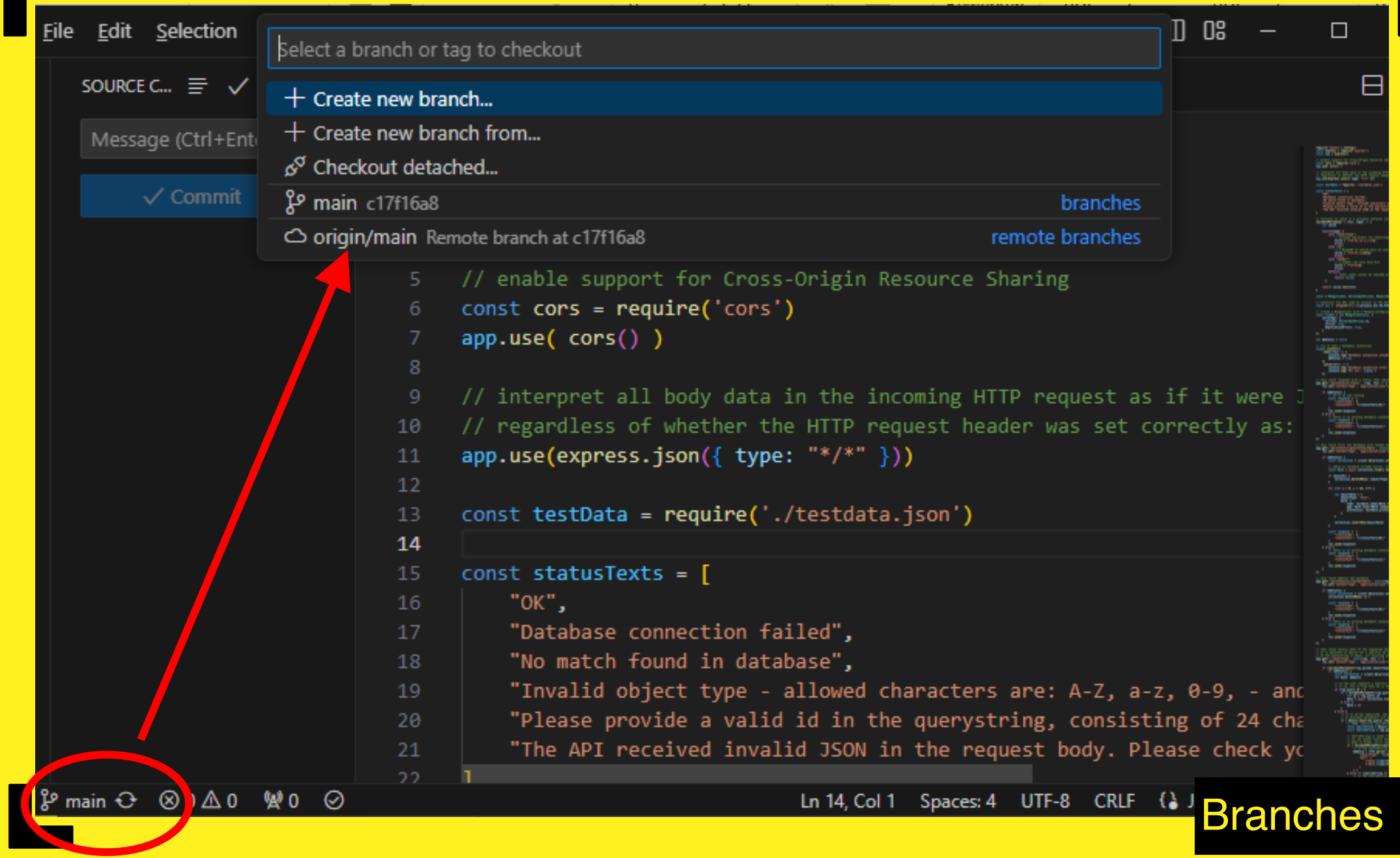
GitHub flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly. This guide explains how and why GitHub flow works.

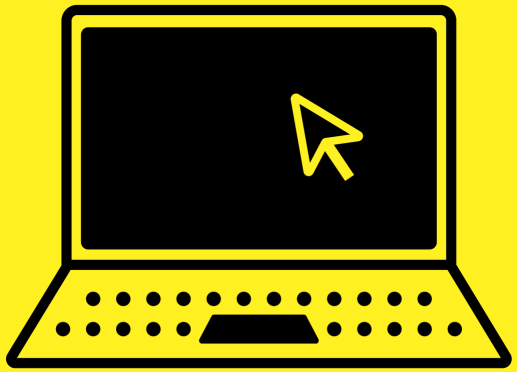


Create a branch

When you're working on a project, you're going to have a bunch of

Branches


File
Edit
Selection
SOURCE C...
Message (Ctrl+Ent
Commit
Select a branch or tag to checkout
Create new branch...
Create new branch from...
Checkout detached...
main c17f16a8
branches
origin/main Remote branch at c17f16a8
remote branches
// enable support for Cross-Origin Resource Sharing
const cors = require('cors')
app.use(cors())
// interpret all body data in the incoming HTTP request as if it were
// regardless of whether the HTTP request header was set correctly as:
app.use(express.json({ type: "*/*" }))
const testData = require('./testdata.json')
const statusTexts = [
"OK",
"Database connection failed",
"No match found in database",
"Invalid object type - allowed characters are: A-Z, a-z, 0-9, - and
"Please provide a valid id in the querystring, consisting of 24 cha
"The API received invalid JSON in the request body. Please check yo
main
Ln 14, Col 1
Spaces: 4
UTF-8
CRLF
Branches



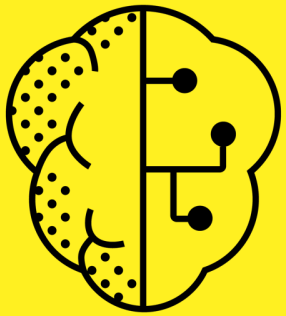
Live demo

Branches & merging

in VS Code

Lesopdracht

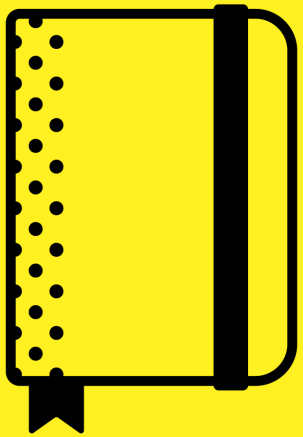
Reviewing Pull Requests & Managing Merge Conflicts



Github video’s & oefeningen

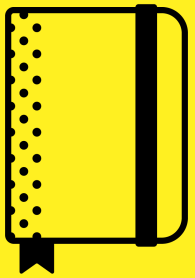
- Github flow and branches
- Branch protection
- Projects and issues
- Samenwerken op Github

Links in lesplanning



Theorie

Responsive

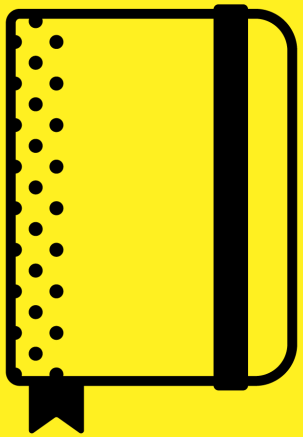


Responsive

Good practices

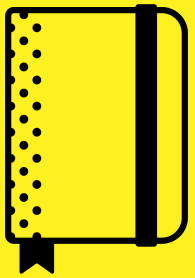
- Relatieve eenheden: em, rem, vh, vw, %
- Media queries: cut-off instellen voor kleinere schermen
- Flexbox: Flexibel gebruik van breedte/hoogte, wrap
- Grid: Automatiseren van kolommen en rijen voor bijv. galleries

Onderzoek welke vorm van responsive design optimaal is voor jouw component/website en pas die toe.



Theorie

Coding standards

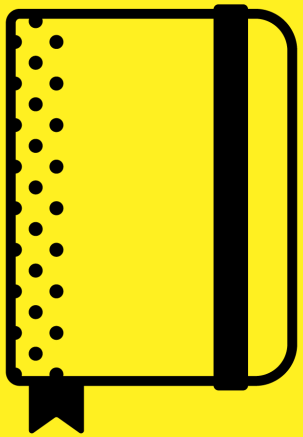


Coding standards

(Team)afspraken over codeerstijl

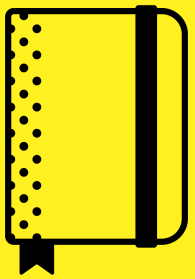
- DTT (Amsterdams digital agency)
- Ryan McDermott: clean code JavaScript
- Google JavaScript style guide

Bekijk de voorbeelden en links in de lesplanning en maak met je team afspraken over jullie codeerstijl.



Theorie

EcmaScript 6



EcmaScript 6

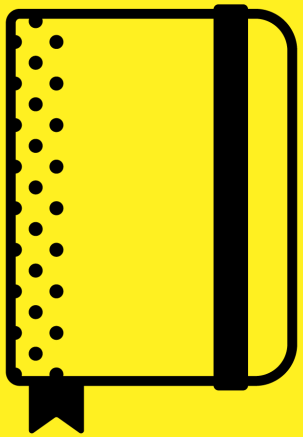
Uitgebreide tutorial in lesplan

Belangrijkste veranderingen

- *let* als vervanging van *var*
- *const* voor constanten
- *Arrow functions*

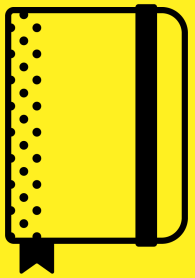
  ```
  (parameter) => { }
  ```
- Puntkomma’s zijn niet meer nodig

  Voor sommigen wel nog steeds voorkeur



Theorie

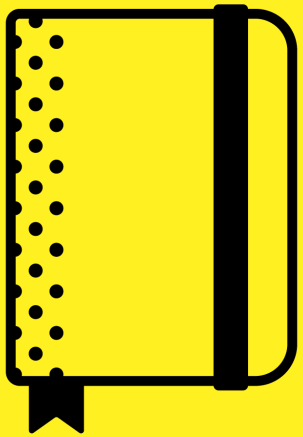
Progressive enhancement



Progressive enhancement

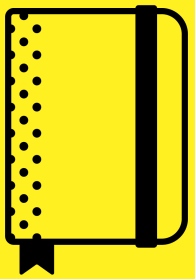
- Bouw functionaliteit stapsgewijs uit
  - HTML > CSS > JavaScript
  - Basisfuncties zijn beschikbaar met enkel HTML
  - CSS en JS bieden extra UX-features
  - Voordeel: JS-bugs kunnen niet de basisfunctionaliteit van de website breken
  - Accessibility is UX!

Doe onderzoek naar de mogelijkheden om jouw component ‘progressive enhanced’ te bouwen.



Theorie

Linters



Linters

Een Linter checkt je source code op mogelijke fouten:

- Helpt bugs voorkomen
- Zorgt dat je code leesbaar blijft voor mede-developers
- Even werk om in te stellen, maar scheelt tijd en stress op lange termijn
- Verbetert de samenwerking in het team

Opdracht: ESLint

1. Welke code standards wil je gebruiken in je project?
2. Installeer en configureer ESLint in je project volgens de instructies in de lesplanning
3. Lint je code!