Progressive Enhancement

A design/architecture philosophy

- Base content should be accessible to all browsers
    - Semantics!
    - No JS
- Extra/Altered layout with sophisticated CSS
- Extra/Altered functionality with JS

Benefits of Progressive Enhancement

- Available to widest possible audience
    - Not everyone is running latest browsers
        - Not everyone CAN run latest browsers
    - Including unknown audiences of the future
    - Such as programs that can't run JS
        - or can't understand visual effects
        - Existing and future
- Potential flexibility when faced with change
    - Strong separation of content and presentation

Graceful Degradation

- Flipside of Progressive Enhancement
    - Same goal though: Maximize availability
- Write for modern browsers
    - Fail back to older standards
        - CSS
            - Older property value, newer value
                - Newer ignored if unknown
            - Feature detection
        - JS
            - Feature detection
            - Polyfills

Progressive Enhancement: Basic How To

- Have the page be usable
    - With no CSS
    - With no JS
    - Requires Semantic HTML
    - Requires meaningful text order
    - Requires working forms/links
- Next, if CSS loads, page is nicer
- Next, if JS loads, page is nicer
    - Ex: Adds form validation
    - Ex: e.preventDefault and changes content

Graceful Degradation: Basic How To

Option 1:

- Use an older, reliable value for property
    - Follow with newer/riskier value for property
    - Browsers will ignore "invalid" values
- This is regularly done for "prefix" values

```
display: flex; /* Added ~2012 */
display: grid; /* Added ~2017 */
```

Option 2:

- `@supports` CSS Feature Queries (~2021) (See MDN)

Cons and Costs of Progressive Enhancement/Graceful Degradation

- Extra Development effort
    - Smaller audience impact
        - No audience impact?
- Extra Testing effort
    - Have and use old browsers?
        - Other Operating Systems?
- Many like the idea, relatively few implement
    - Big companies
    - Focused publishers

Use of Progressive Enhancement is not a binary

- Can use partially
    - Particularly when adopting newer features
- Can create MVP (Minimal Viable Product)
    - Make sure core features work for all
    - Nicest options for latest
    - May be done as separate app
        - Ex: Gmail "Basic HTML" option

Accessibility best done at start

- Don't think you can come back later and "add" it
  - Then it will be "rewrite", not "add"
- Don't become inaccessible by accident
  - Delaying considering = choosing inaccessibility
- Accessibility has many benefits
  - Decide what effort to provide
  - That decision can save a lot of time/$ later!
  - Costs mostly come from delayed consideration