CSS Internals

http://nathankrasney.com/

Three principles for writing css+html

1. **Responsive design** : fluid layouts (grid ? , flex?) , media queries , responsive images , correct units , desktop first vs mobile first
2. **Maintainable and scalable code** : clean , easy to understand , growth , reuseable , how to organize the files , how to name the classes , how to structure the HTML
3. **Web performance** : less http request , less code , compress code , use css preprocessor , less images , compress images

Responsive design principles

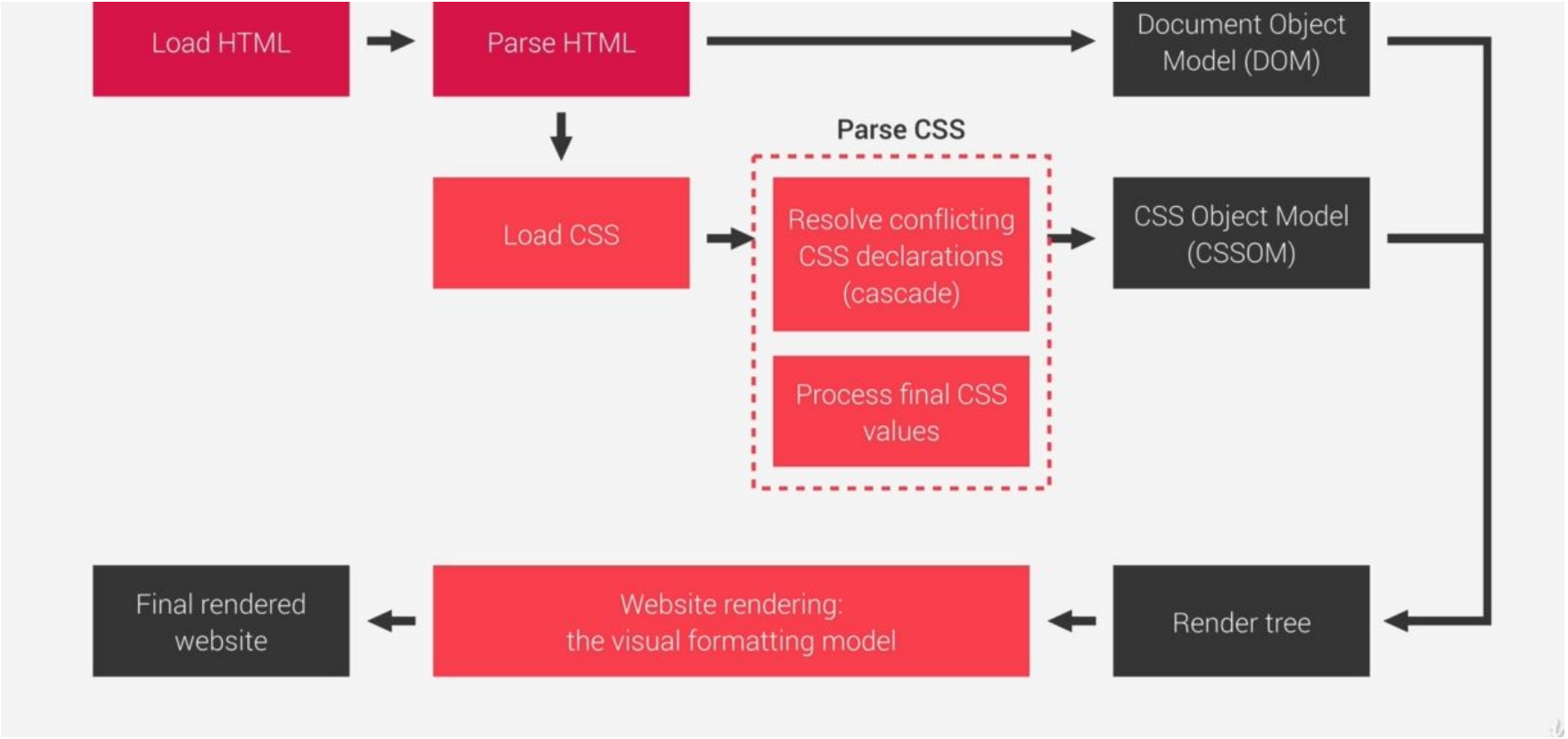
1. **Fluid grids and layout**
   a. use % for width
   b. Use float , grid , flex

2. **Responsive images**

3. **Media queries :** change style on breakpoints

http://nathankrasney.com/

What happen when html file is loaded to the browser



http://nathankrasney.com/

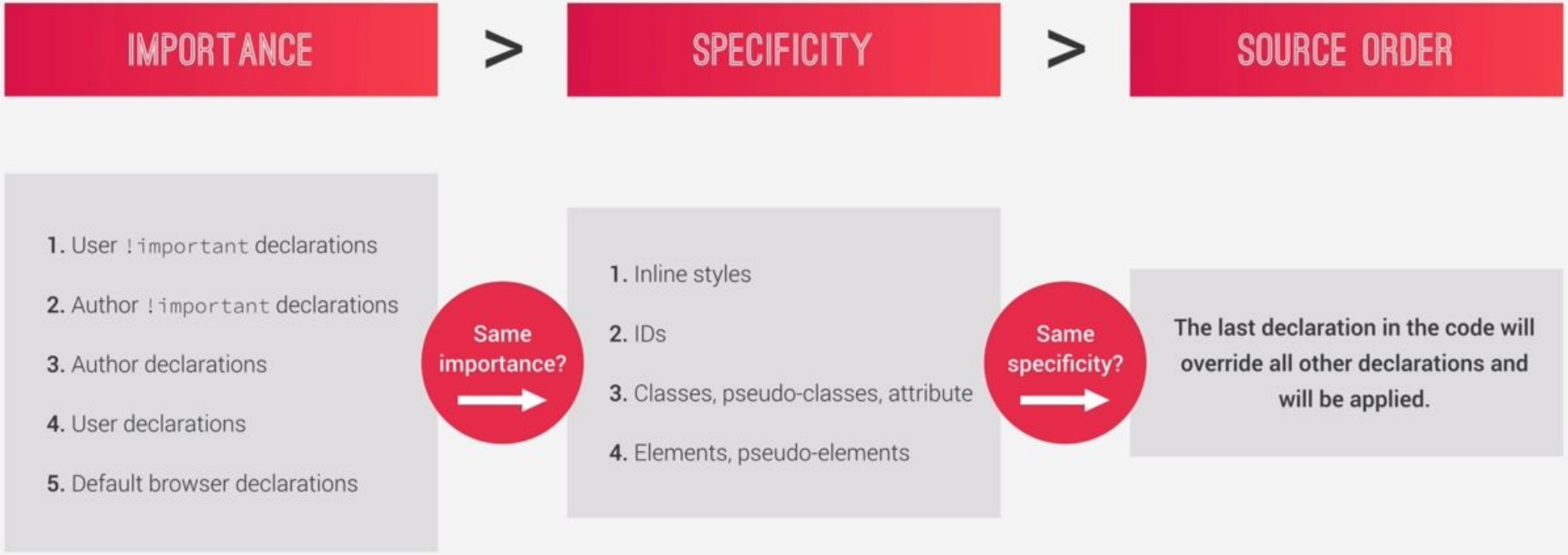
Cascade

Cascade : process of combining different stylesheets and resolving conflicts between different css rules and declarations e.g. when more than one rule applies to the same element

Css sources :

- **Author** - refer to developers which create styles
- **User** - simple user changing default font in the browser
- **Browser (user agent)** - the default browser styling e.g. default blue color for anchor

How css cascade conflicts are resolved



http://nathankrasney.com/

Css parsing - values

	width (paragraph)	padding (paragraph)	font-size (root)	font-size (section)	font-size (paragraph)
1. Declared value (author declarations)	140px 66%	–	–	1.5rem	–
2. Cascaded value (after the cascade)	66%	–	16px (Browser default)	1.5rem	–
3. Specified value (defaulting, if there is no cascaded value)	66%	0px (Initial value)	16px	1.5rem	24px
4. Computed value (converting relative values to absolute)	66%	0px	16px	24px (1.5 * 16px)	24px
5. Used value (final calculations, based on layout)	184.8px	0px	16px	24px	24px
6. Actual value (browser and device restrictions)	185px	0px	16px	24px	24px

INHERITANCE

```html
<div class="section">
  <p class="amazing">CSS is absolutely amazing</p>
</div>
```

```css
.section {
  font-size: 1.5rem;
  width: 280px;
  background-color: orangered;
}

p {
  width: 140px;
  background-color: green;
}

.amazing {
  width: 66%;
}
```

(Let's analyse the green paragraph)

http://nathankrasney.com/

Css parsing - units

		Example (x)	How to convert to pixels	Result in pixels
	% (fonts)	150%	x% * parent's computed font-size	24px
	% (lengths)	10%	x% * parent's computed **width**	100px
Font-based	em (font)	3em	x * **parent** computed font-size	72px (3 * 24)
Font-based	em (lengths)	2em	x * **current element** computed font-size	48px
Font-based	rem	10rem	x * **root** computed font-size	160px
Viewport-based	vh	90vh	x * 1% of viewport height	90% of the current viewport height
Viewport-based	vw	80vw	x * 1% of viewport width	80% of the current viewport width

```
html, body {
  font-size: 16px;
  width: 80vw;
}

header {
  font-size: 150%;
  padding: 2em;
  margin-bottom: 10rem;
  height: 90vh;
  width: 1000px;
}

.header-child {
  font-size: 3em;
  padding: 10%;
}
```

http://nathankrasney.com/

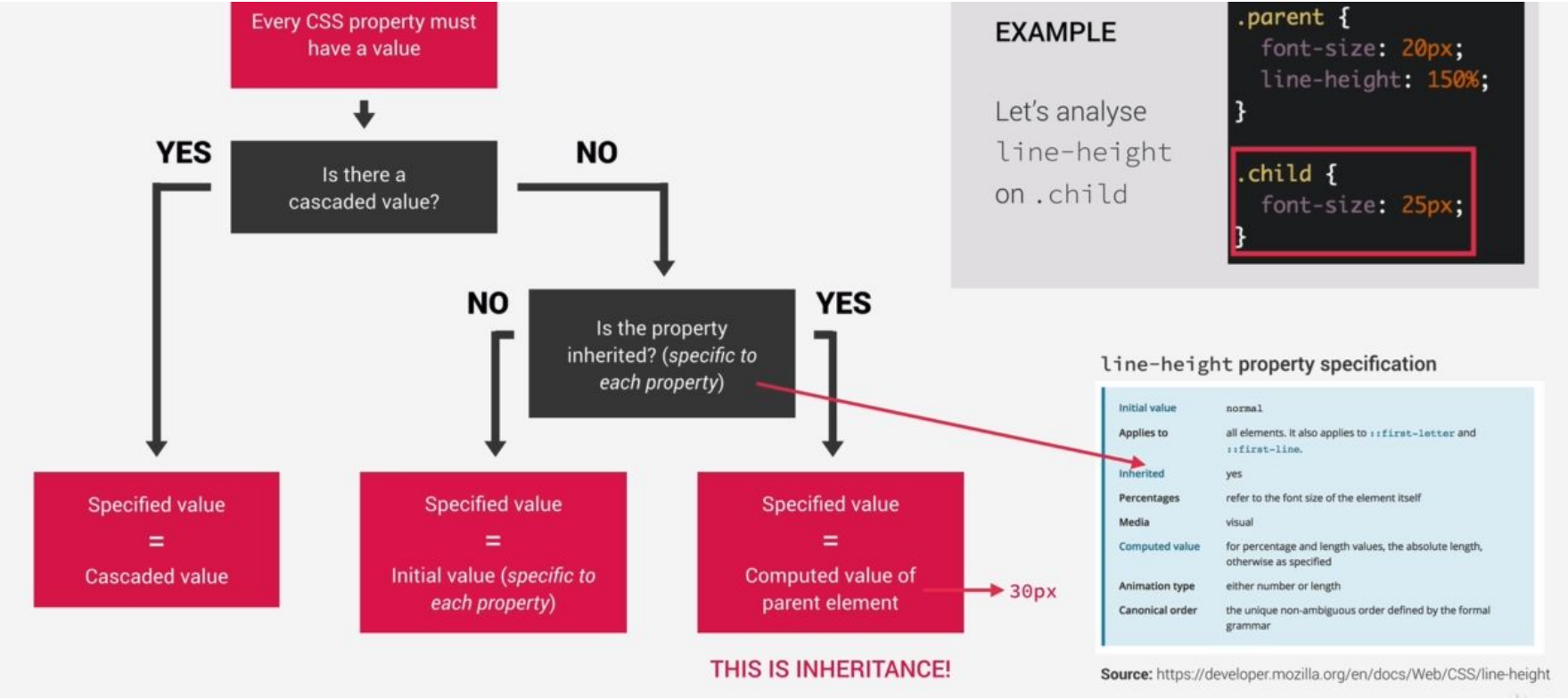
Css parsing - summary

- Each property has an initial value, used if nothing is declared (and if there is no inheritance — see next lecture);
- Browsers specify a **root `font-size`** for each page (usually `16px`);
- Percentages and relative values are always converted to pixels;
- Percentages are measured relative to their parent's **`font-size`**, if used to specify `font-size`;
- Percentages are measured relative to their parent's **`width`**, if used to specify lengths;
- `em` are measured relative to their **parent** `font-size`, if used to specify `font-size`;
- `em` are measured relative to the **current** `font-size`, if used to specify lengths;
- `rem` are always measured relative to the **document's root** `font-size`;
- `vh` and `vw` are simply percentage measurements of the viewport's `height` and `width`.

http://nathankrasney.com/

Css parsing - inheritance

Inheritance - how to propagate parent element properties to child element



http://nathankrasney.com/

Inheritance - summary

- Inheritance passes the values for some specific properties from parents to children — **more maintainable code**;
- Properties related to text are inherited: `font-family`, `font-size`, `color`, etc;
- The computed value of a property is what gets inherited, **not** the declared value.
- Inheritance of a property only works if no one declares a value for that property;
- The `inherit` keyword forces inheritance on a certain property;
- The `initial` keyword resets a property to its initial value.

http://nathankrasney.com/

The visual formatting model (check 3rd slide)

DEFINITION

Algorithm that calculates boxes and determines the layout of theses boxes, for each element in the render tree, in order to determine the final layout of the page.

- **Dimensions of boxes**: the box model;
- **Box type**: inline, block and inline-block;
- **Positioning scheme**: floats and positioning;
- **Stacking contexts**;
- Other elements in the render tree;
- Viewport size, dimensions of images, etc.

http://nathankrasney.com/

BEM (naming conventions)

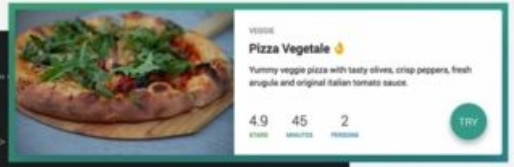
BEM

- **B**lock **E**lement **M**odifier
- **BLOCK**: standalone component that is meaningful on its own.
- **ELEMENT**: part of a block that has no standalone meaning.
- **MODIFIER**: a different version of a block or an element.

```
.block {}
.block__element {}
.block__element--modifier {}
```

Low-specificity BEM selectors

```
<figure class="recipe">
  <div class="recipe__hero">
    <img class="recipe__img" src="
  </div>
  <div class="recipe__info">
    <div class="recipe__category">
      Veggie
    </div>
    <figcaption class="recipe__details">
      <h2 class="recipe__title">Pizza Vegetale 👌</h2>
      <p class="recipe__description">
        Yummy veggie pizza with tasty olives
      </p>
    </figcaption>
    <div class="recipe__stats-box">
      <div class="recipe__stat">
        <span class="recipe__stat-value">4.9</span>
        <span class="recipe__stat-name recipe__stat-name--1">Stars</span>
      </div>
      <div class="recipe__stat">
        <span class="recipe__stat-value">45</span>
        <span class="recipe__stat-name recipe__stat-name--2">Minutes</span>
      </div>
      <div class="recipe__stat">
        <span class="recipe__stat-value">2</span>
        <span class="recipe__stat-name recipe__stat-name--3">Persons</span>
      </div>
    </div>
  </div>
  <a class="recipe__btn btn btn--round" href="#">Try</a>
</figure>
```



http://nathankrasney.com/

Project css directories

Is this structure relevant to sass only ???

This structure is in general for large multi page web sites not for a simple landing page

THE 7 FOLDERS

- base/
- components/
- layout/
- pages/
- themes/
- abstracts/
- vendors/

Global definitions of our entire project like font , reset , animations , utilities , ...

Global header , footer , ...

variables,functions,mixin

Independent and reusable building blocks

Home page , About page , ...

Third parties e.g. bootstap

http://nathankrasney.com/

Css variables

You can use css variable :

- using saas .con : you do need to transpile it to css)
- Using custom properties as done here - excellent video series , here - w3c , here mozilla . pros :
  - you do not need to transpile it to css - this is not possible with saas
  - You can change the css variable on chrome dev tools and see its effect - this is not possible with saas
  - Can be override inside media queries - this is not possible with saas

Responsive images

Screen on pc and mobile do have different screen size , resolution .The same image might not fit all of them .

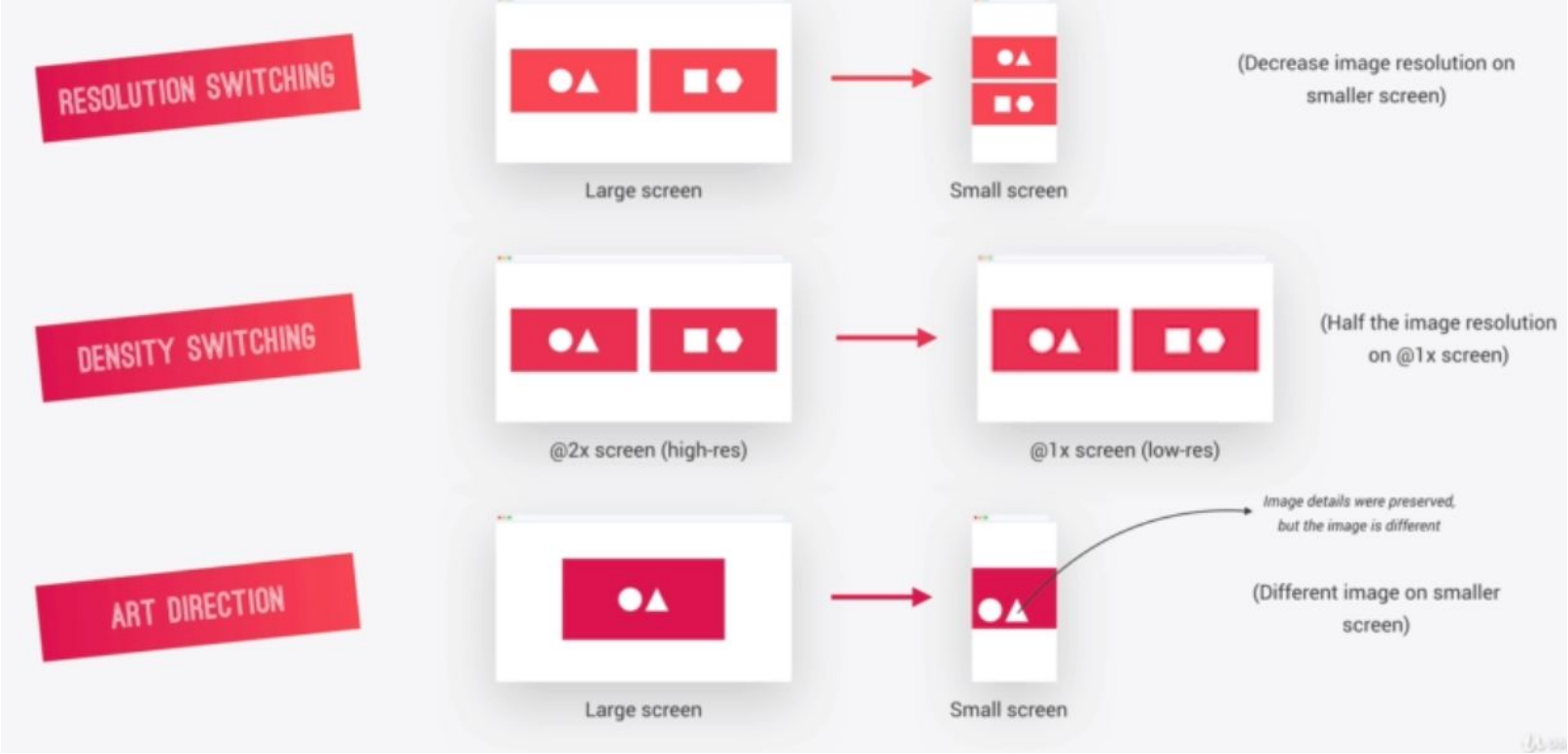
Check the following page on pc and on mobile e.g. iphone 5 (use chrome dev tools) . You will see that the header picture does not look good on mobile

Responsive image means serving the right image for the right screen size.

Check Jonas section 6 , mozilla

http://nathankrasney.com/

Responsive images - use cases



http://nathankrasney.com/