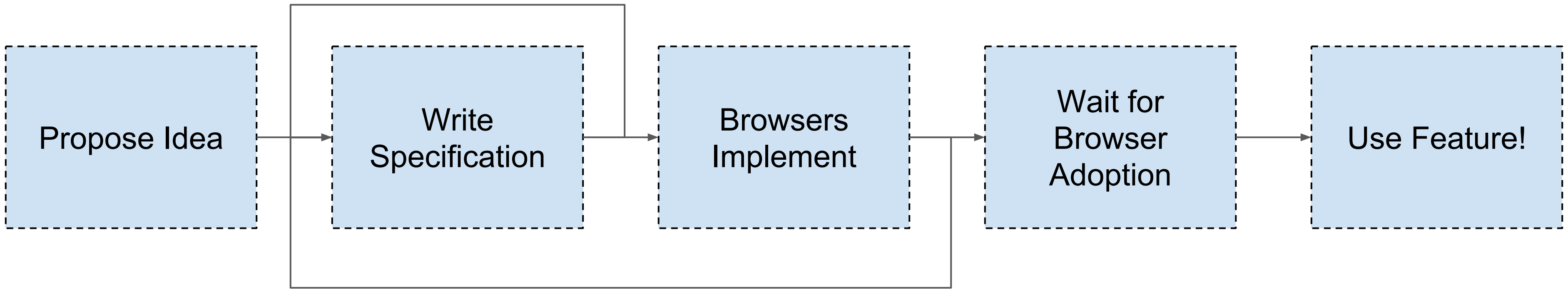
Houdini - Explaining CSS

Ian Kilpatrick - Google Software Engineer

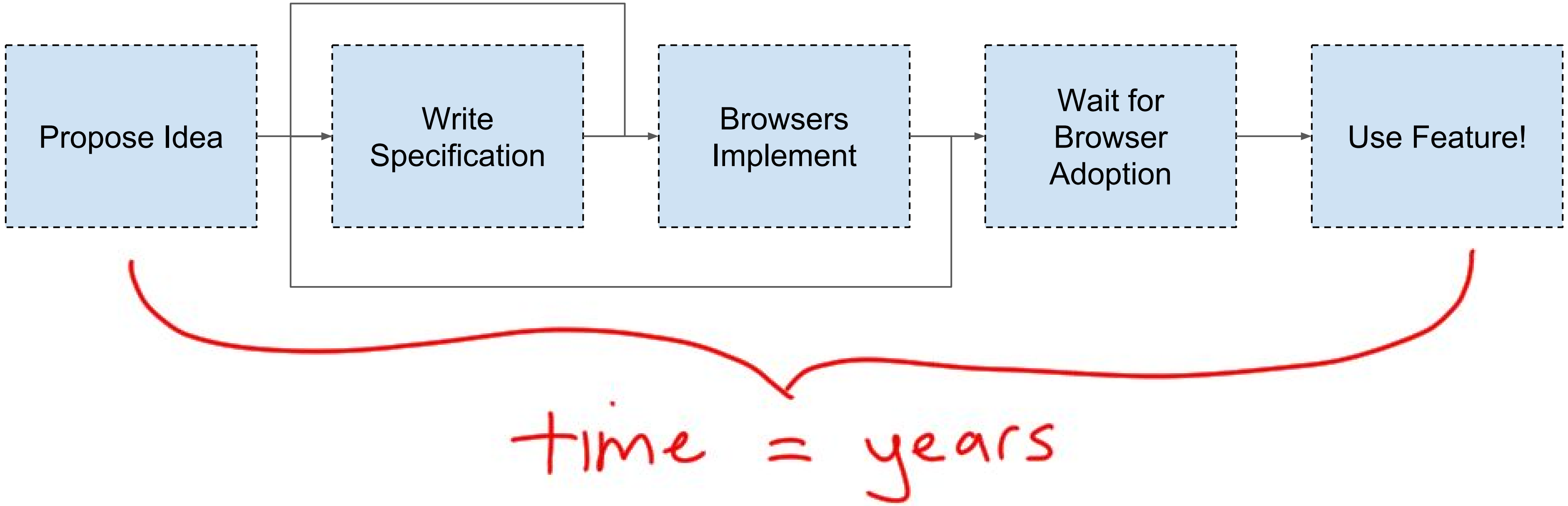
Twitter: @bfgeek

I want to add a feature to CSS

Standards Track

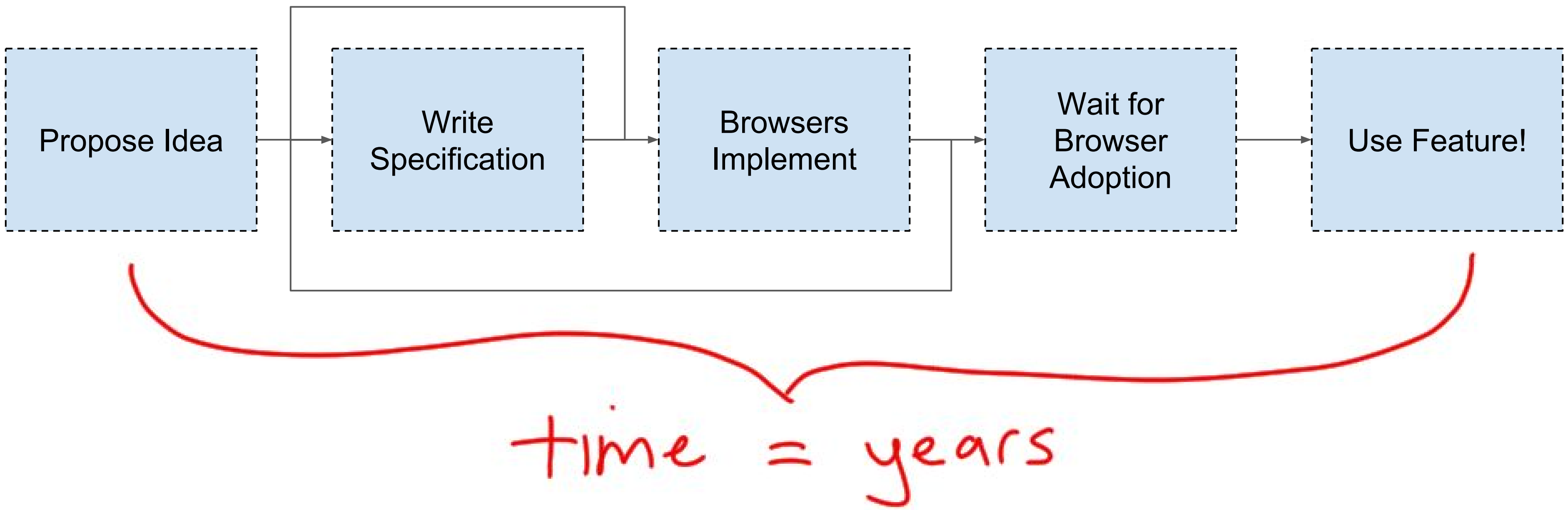


Standards Track

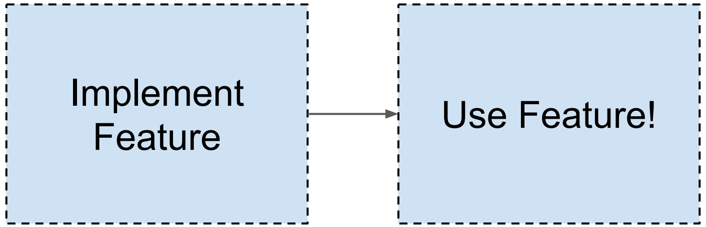


Standards Track

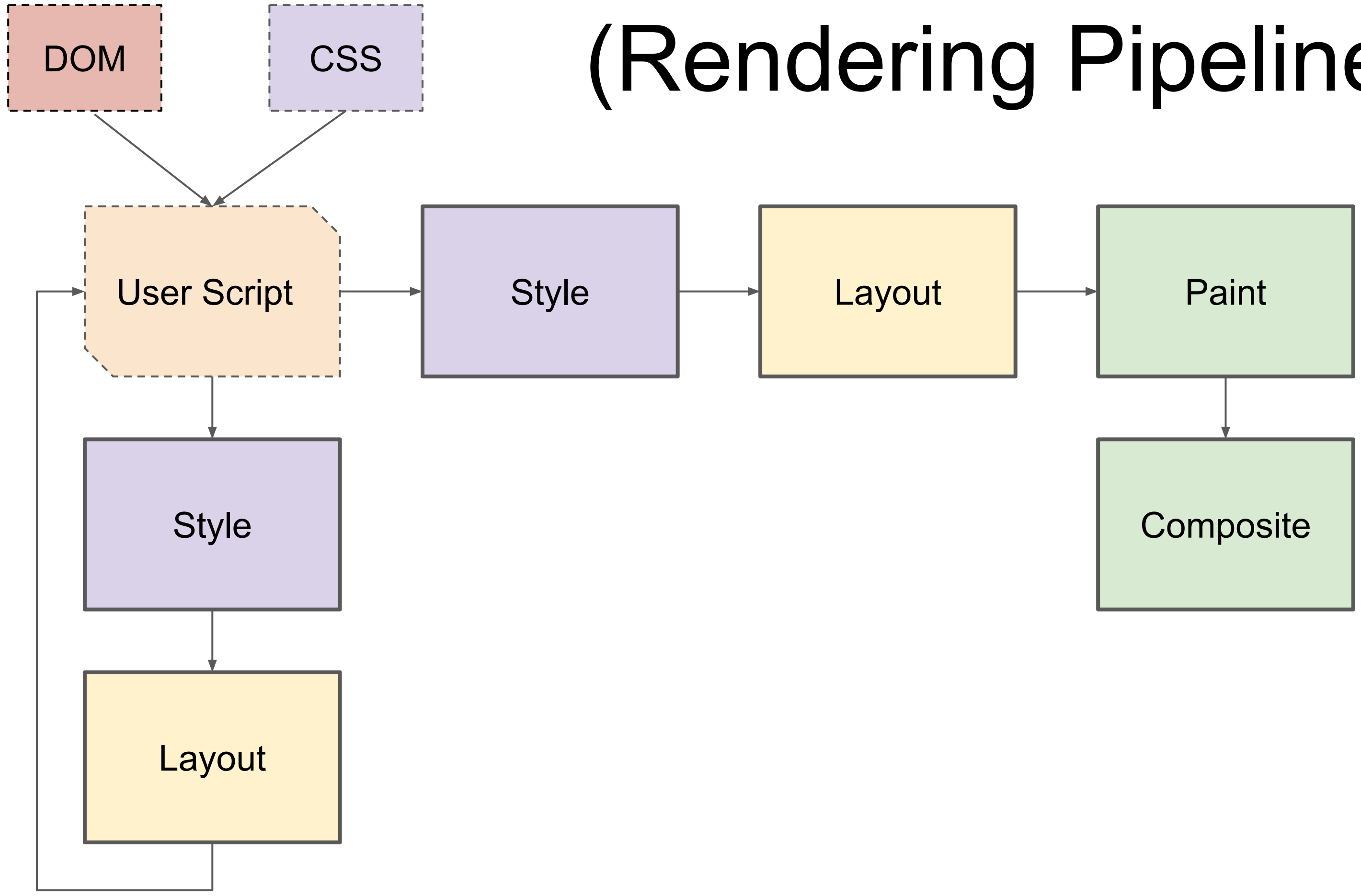
flexbox → first proposed 2009
flexbox → widespread adoption 2014

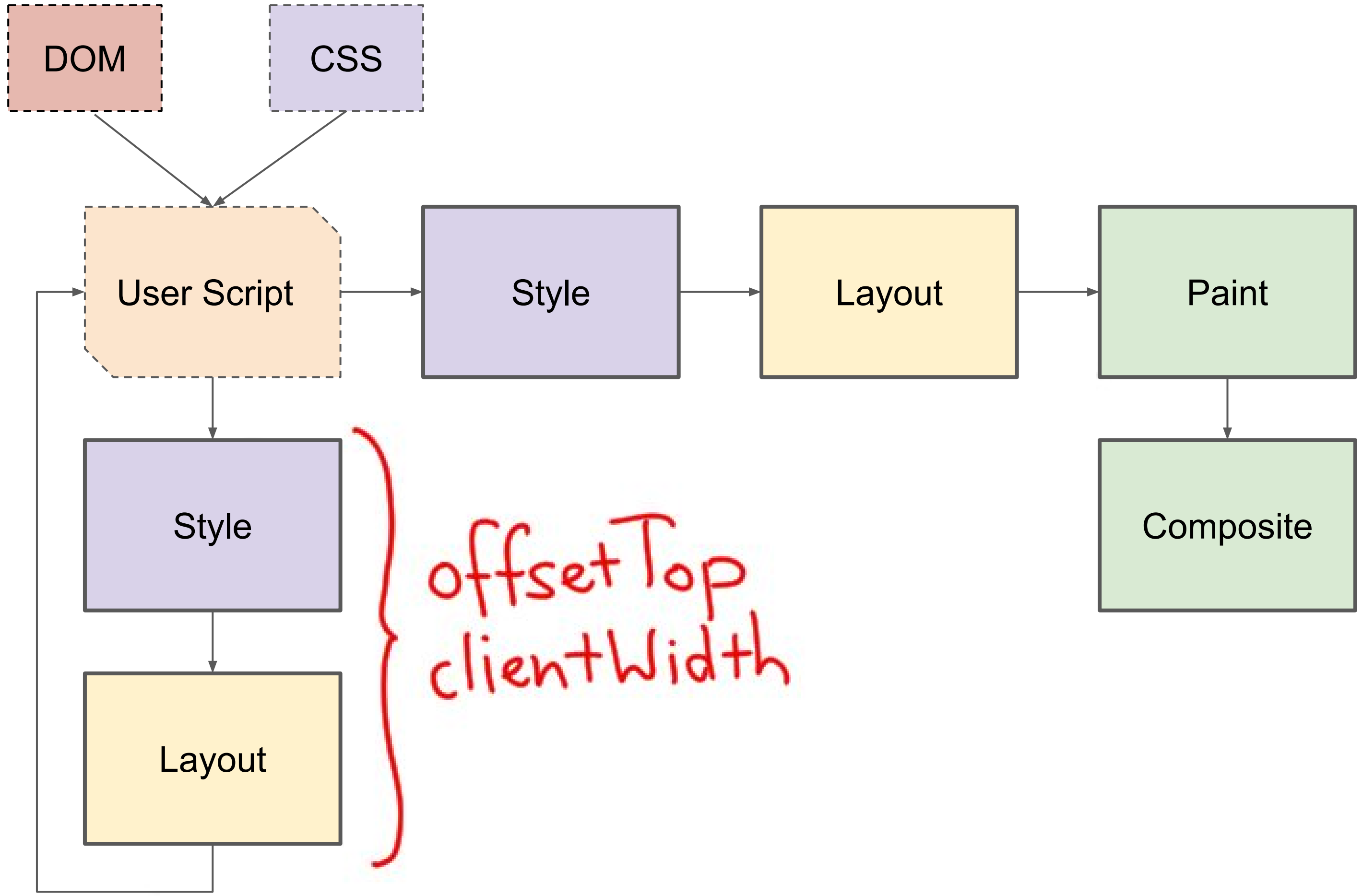


Polyfill Track

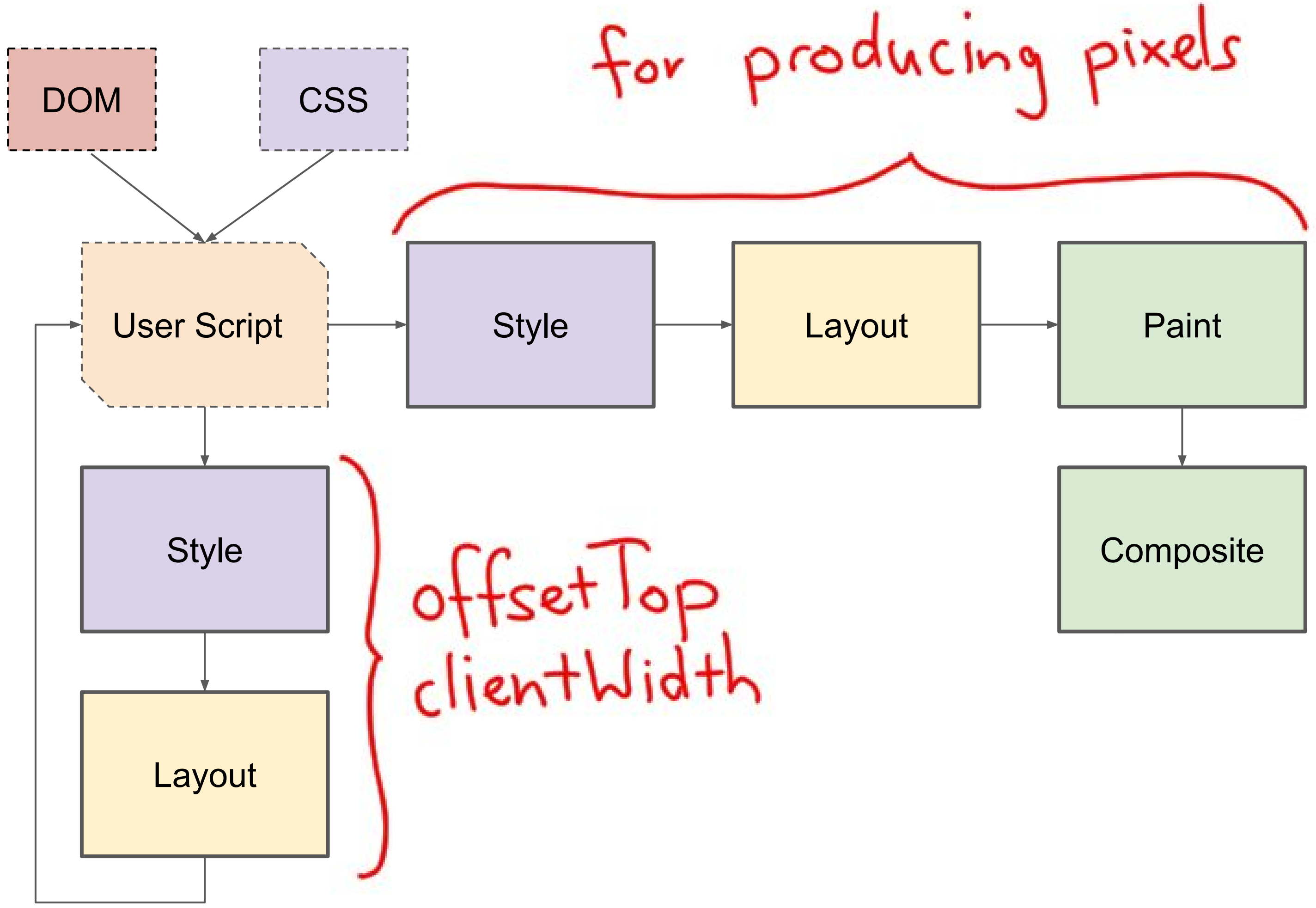


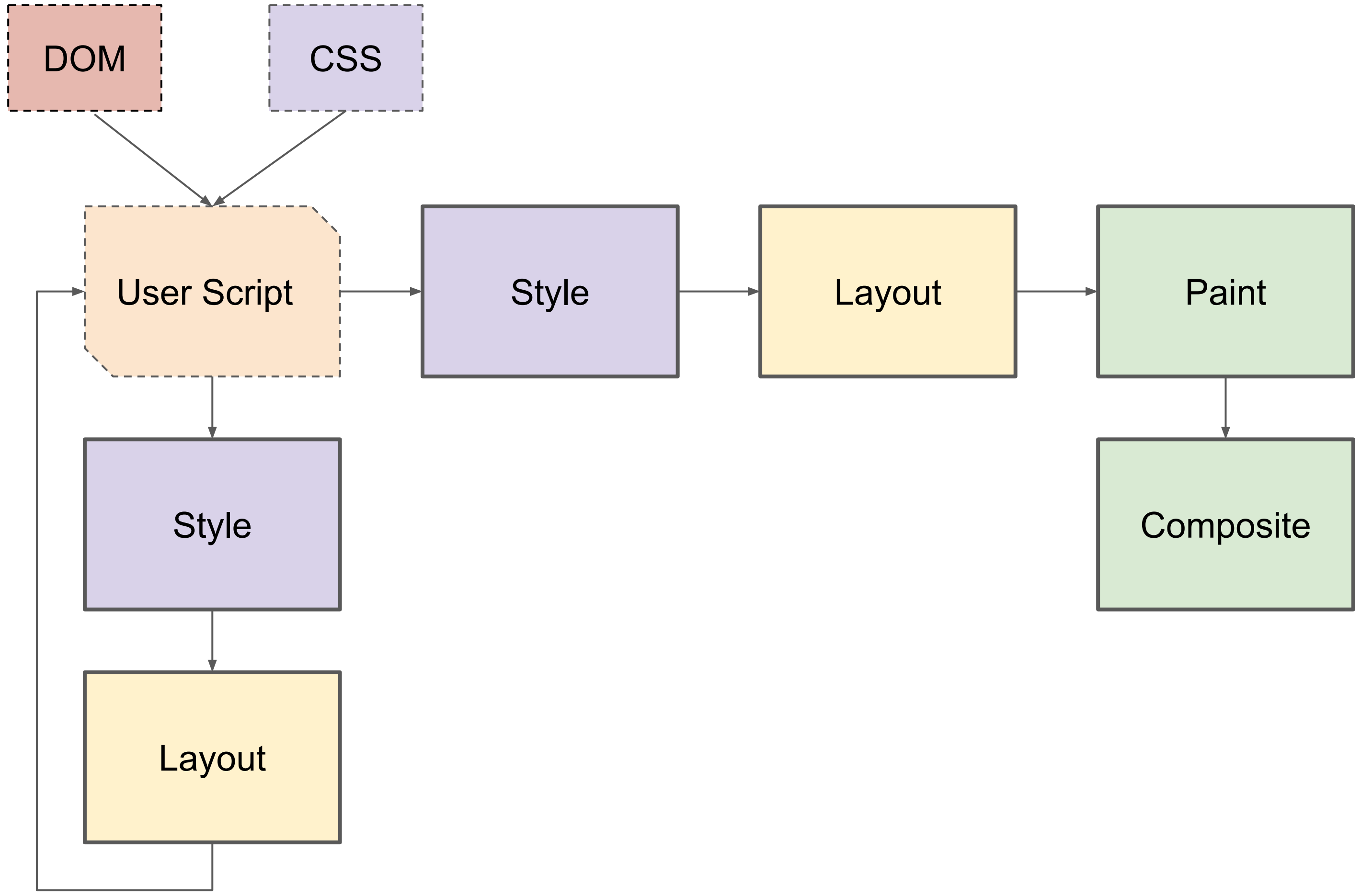
(Rendering Pipeline)

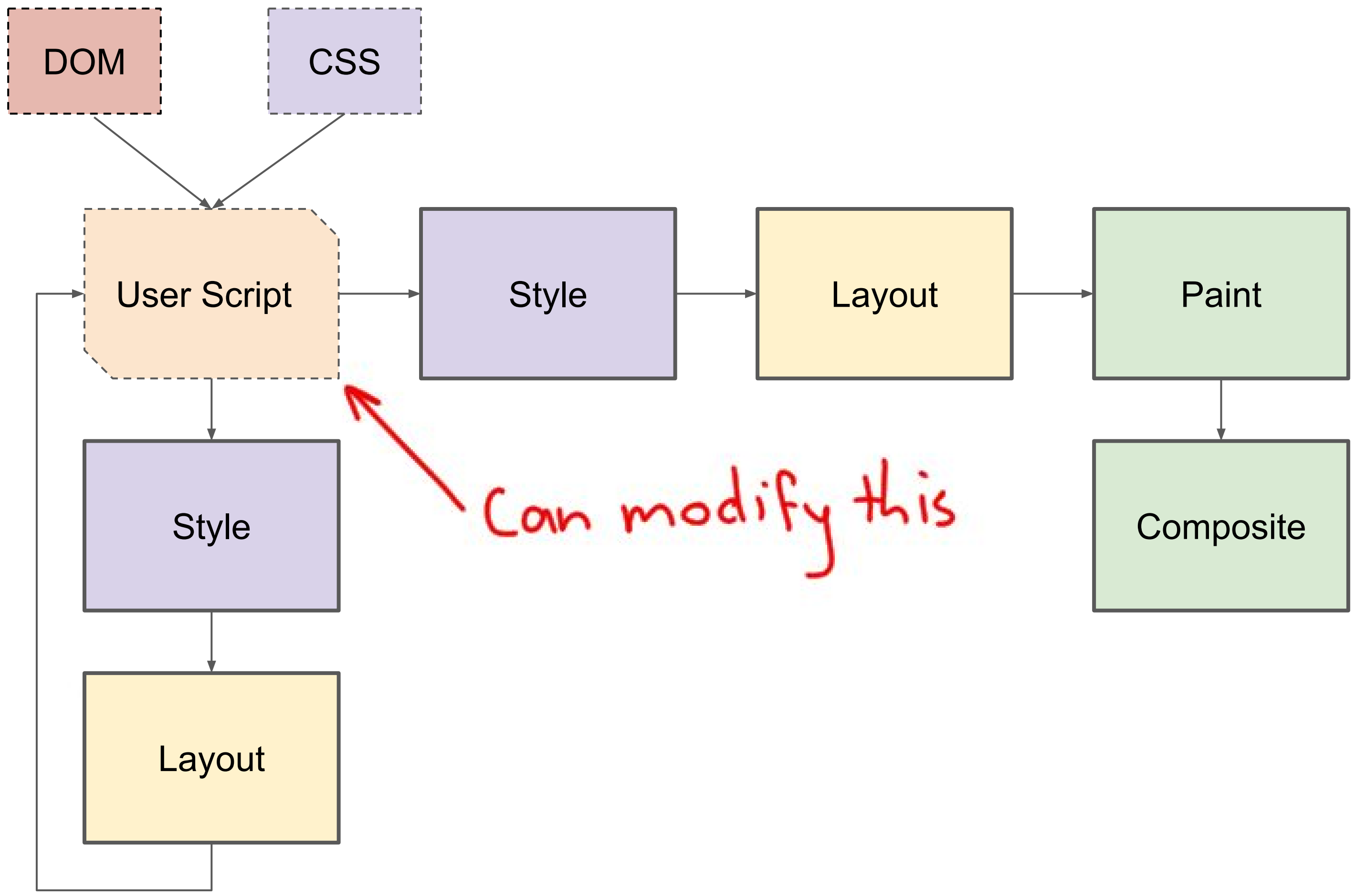


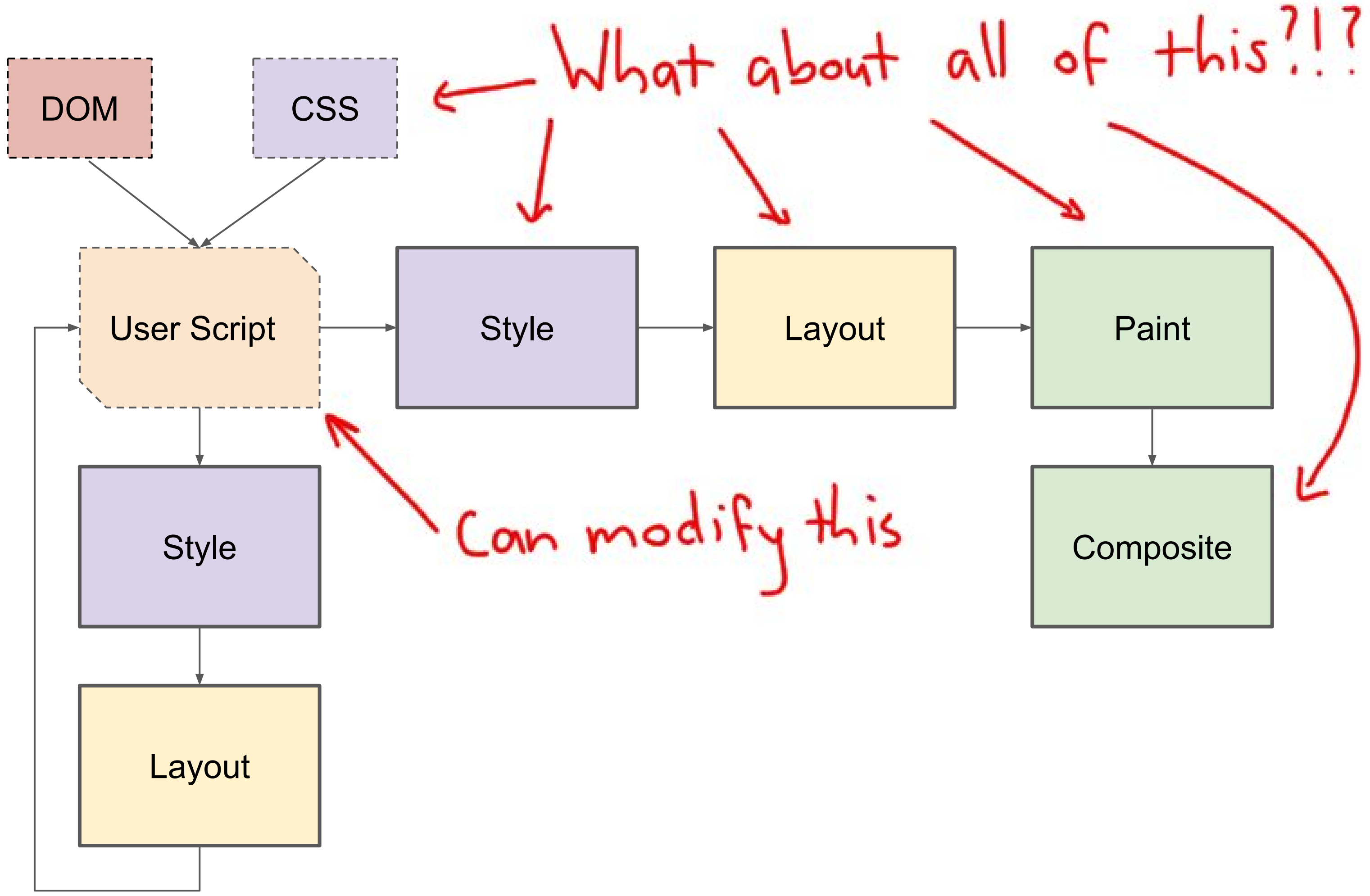


DOM
CSS
User Script
Style
Layout
Paint
Style
Layout
Composite
offsetTop
clientWidth


for producing pixels
DOM
CSS
User Script
Style
Layout
Paint
Composite
Style
Layout
offsetTop
clientWidth


DOM
CSS
User Script
Style
Layout
Paint
Style
Composite
Layout


DOM
CSS
User Script
Style
Layout
Paint
Style
Composite
Layout
Can modify this


What about all of this?!?
DOM
CSS
User Script
Style
Layout
Paint
Composite
Style
Layout
Can modify this

OK, can I add my own CSS properties?

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
}
```

"Initial" value, applies to all elements

```
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
}
```

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
}
```

"Initial" value, applies to all elements

Overrides initial value.

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
}
```

"Initial" value, applies to all elements

Overrides initial value.

Substitutes into scale

"Initial" value, applies to all elements

```css
:root {
  --my-scale: 1;
}
```

Overrides initial value.

```css
.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
}
```

scale(2)

Substitutes into scale

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
  --my-scale: ‘foo’;
}
```

```
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
  --my-scale: ‘foo’;
}
```

Oh noes!

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
  --my-scale: ‘foo’;
}
```

Not actually a number.
A "token stream"
(think of it as a string)

Oh noes!

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
  transition: --my-scale 4s;
}
```

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
  transition: --my-scale 4s;
}
```

Oh noes!

```css
:root {
  --my-scale: 1;
}

.className {
  --my-scale: 2;
  transform: scale(var(--my-scale));
  transition: --my-scale 4s;
}
```

The browser doesn't know how to animate a "token stream".

Oh noes!

```
document.registerProperty({
  name: ‘--my-scale’,
  syntax: ‘<number>’,
  inherits: false,
  initial: ‘1’,
});
```

```
document.registerProperty({
  name: ‘--my-scale’,
  syntax: ‘<number>’,
  inherits: false,
  initial: ‘1’,
});
```

Validate parsing

Animatable

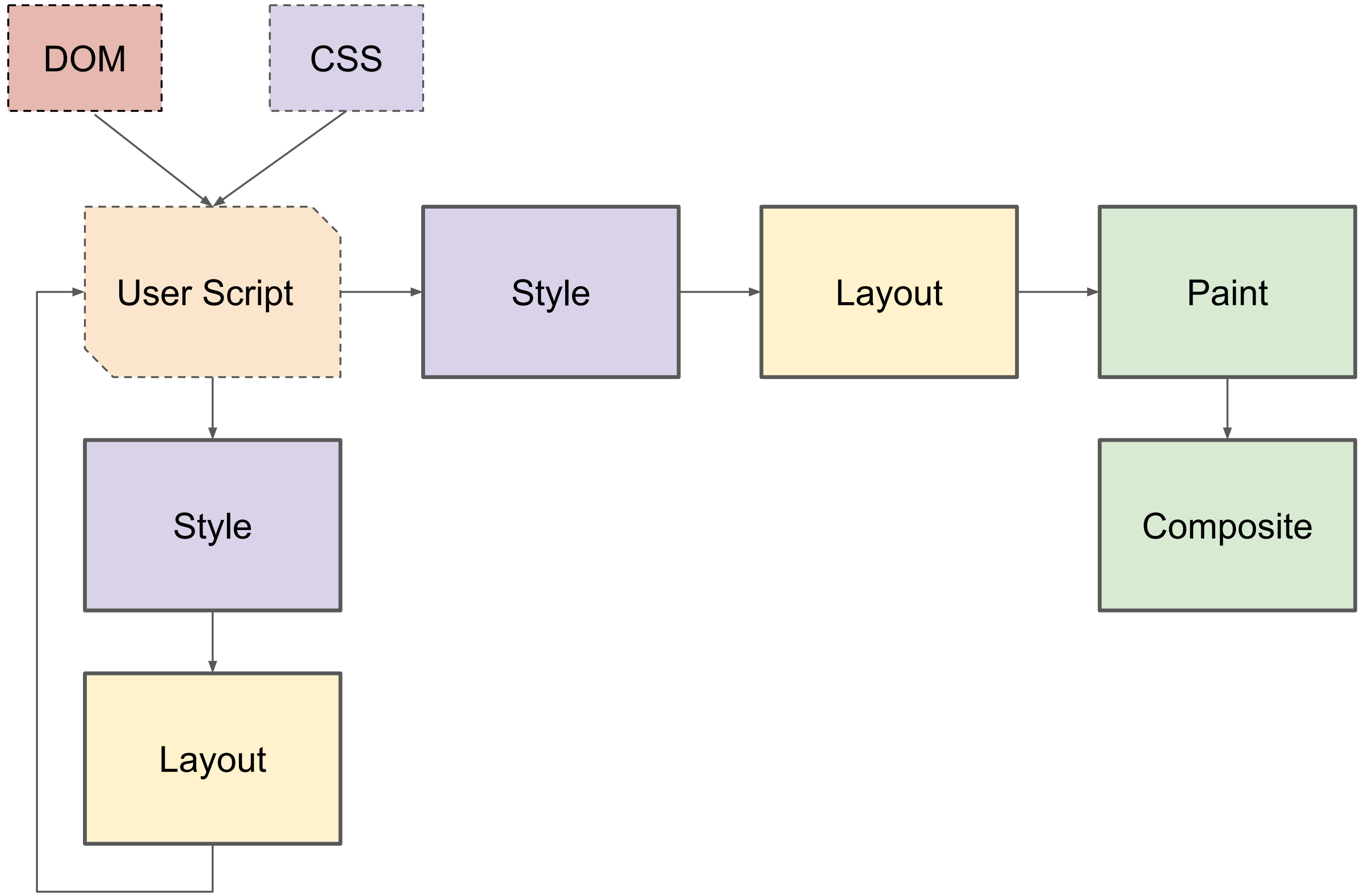
```
document.registerProperty({
  name: ‘--my-scale’,
  syntax: ‘<number>’,
  inherits: false,
  initial: ‘1’,
});
```

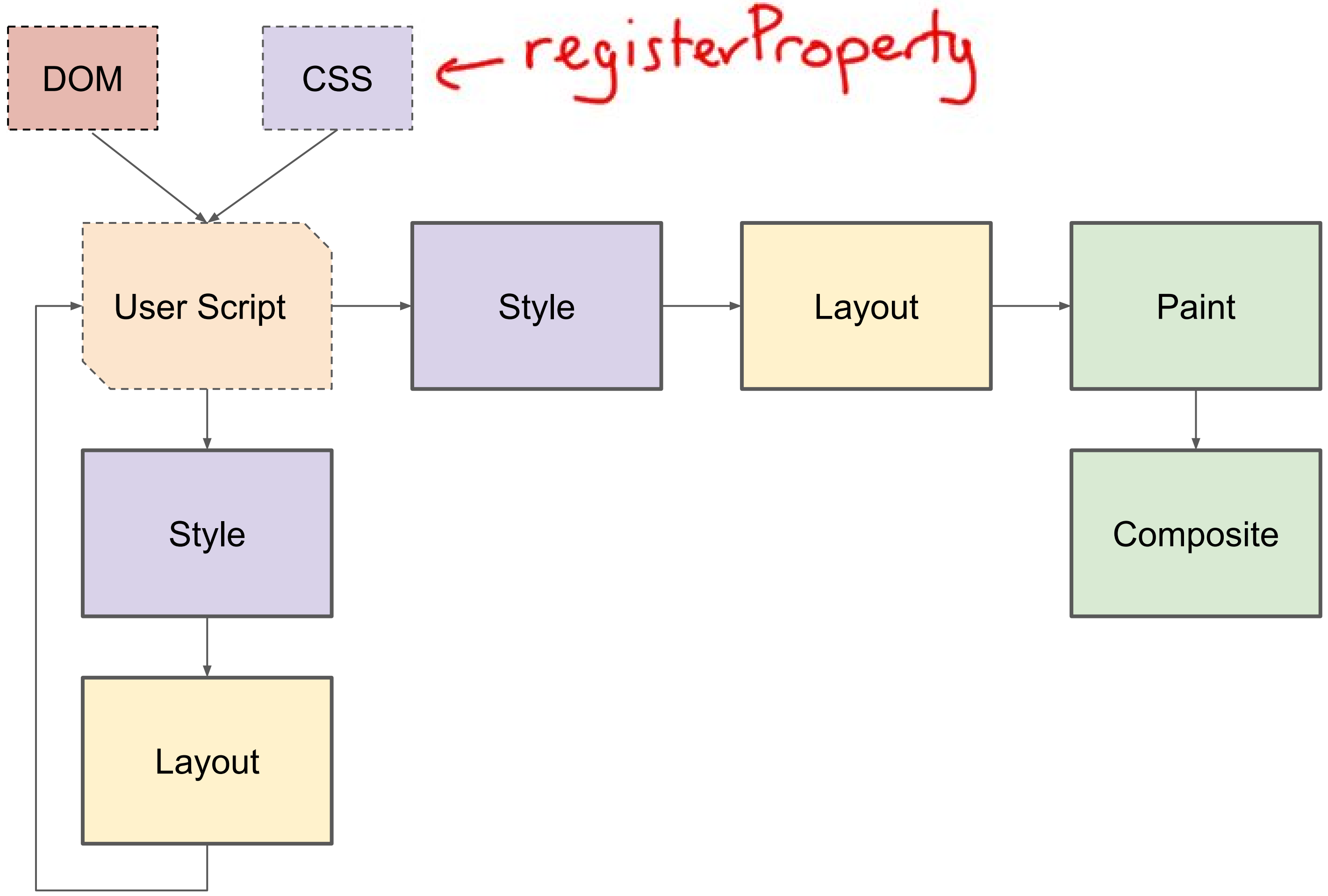
Validate parsing

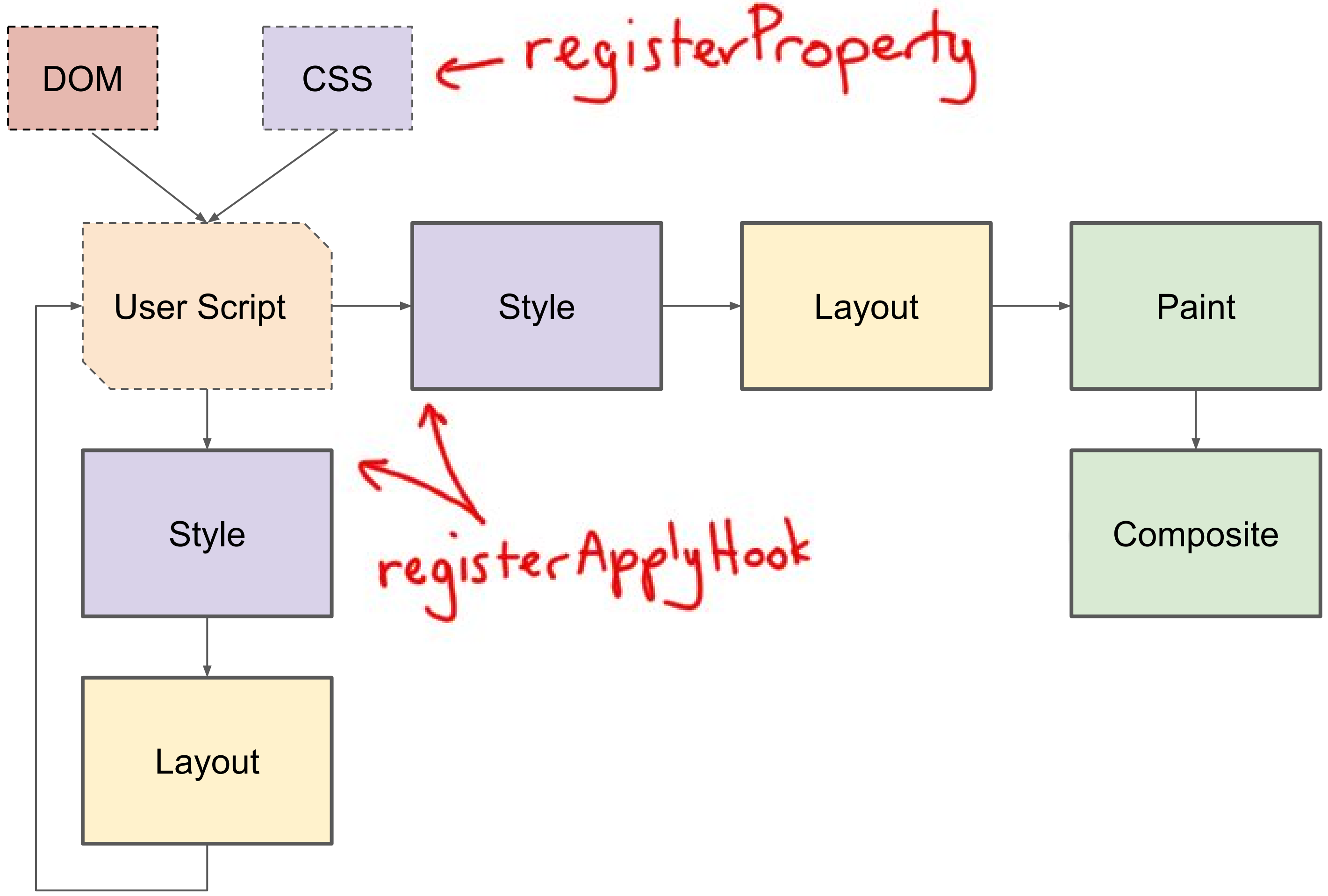
Animatable

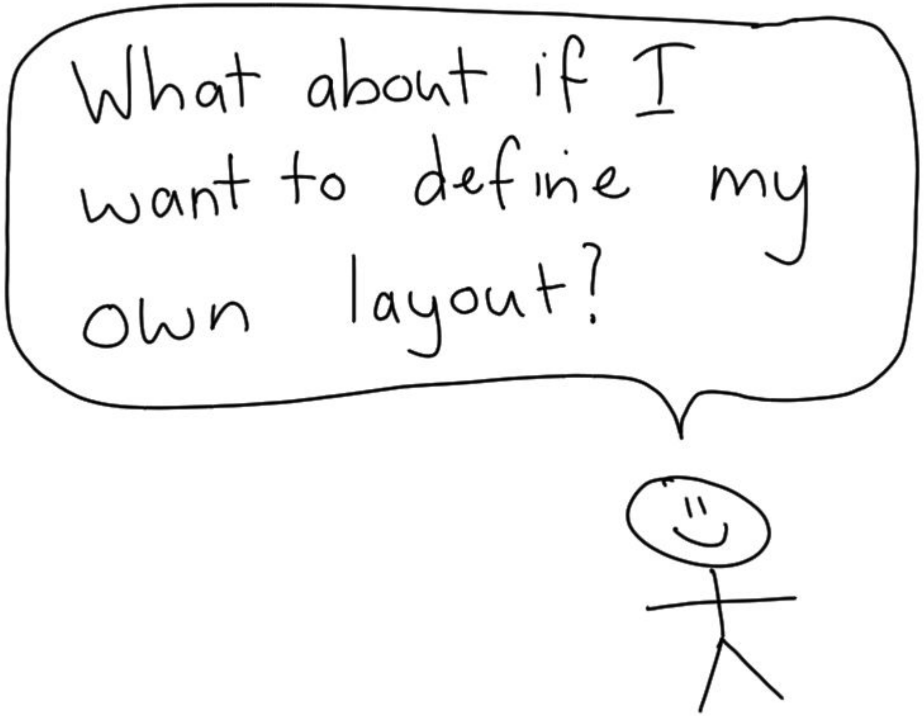
Initial/Default value

```
.className {
  --my-scale: 2;
  --my-scale: ‘foo’;
  transform: scale(var(--my-scale));
  transition: --my-scale 4s;
}
```

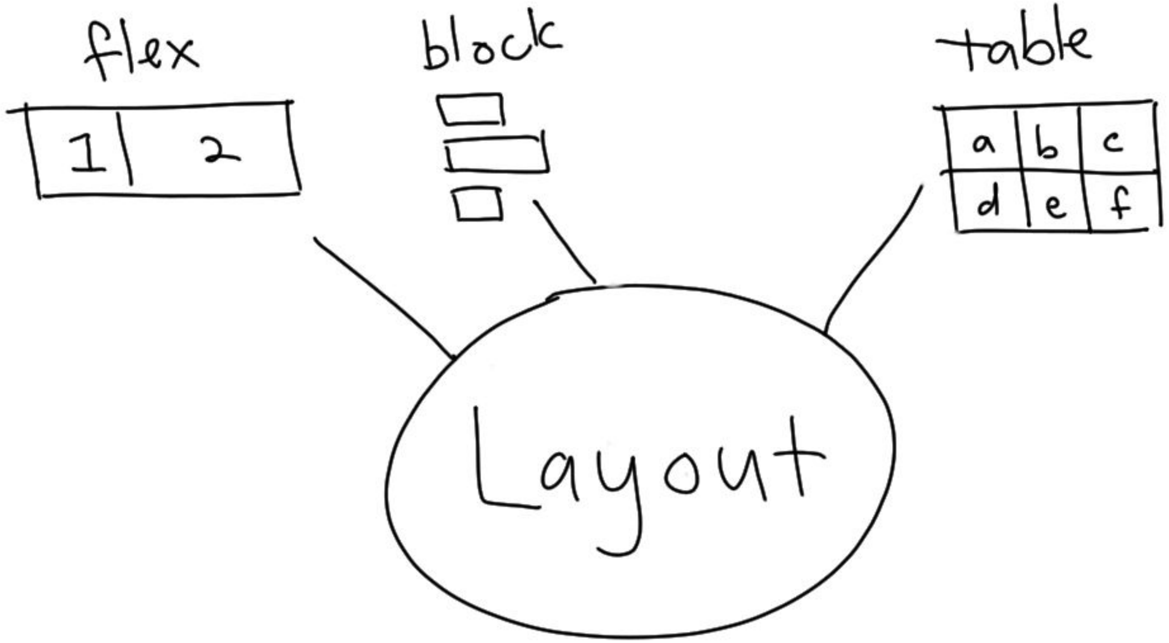

DOM
CSS
User Script
Style
Layout
Paint
Style
Composite
Layout

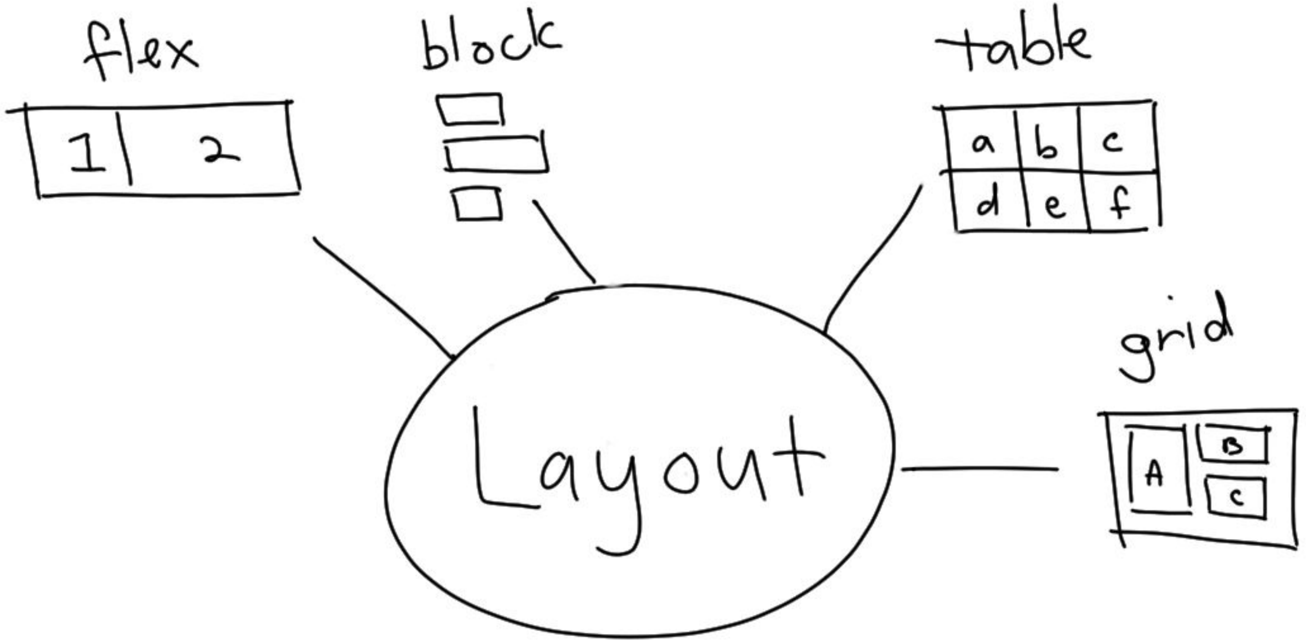

DOM
CSS
← registerProperty
User Script
Style
Layout
Paint
Style
Composite
Layout

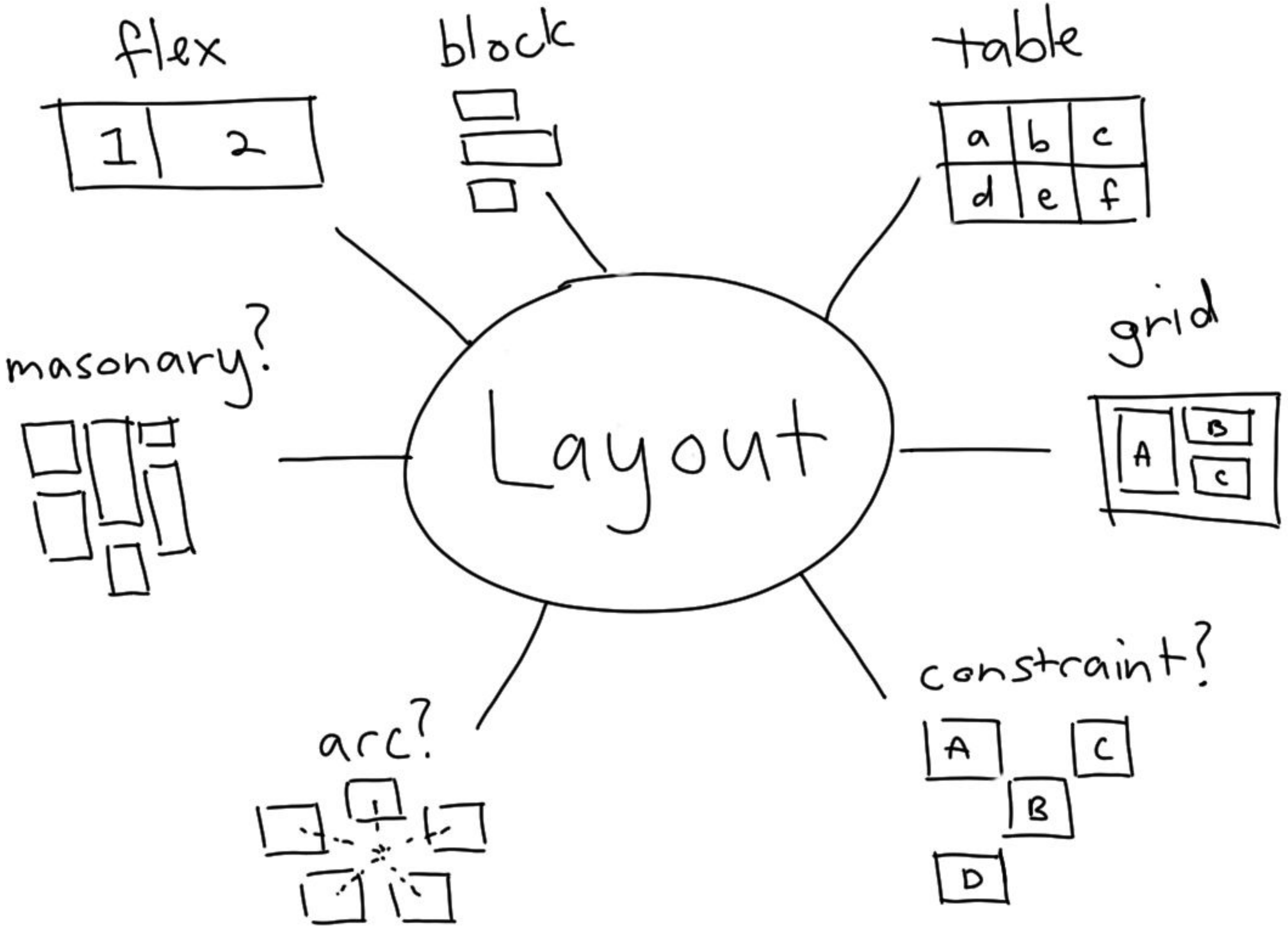

DOM
CSS
registerProperty
User Script
Style
Layout
Paint
Style
registerApplyHook
Composite
Layout


What about if I want to define my own layout?

Layout

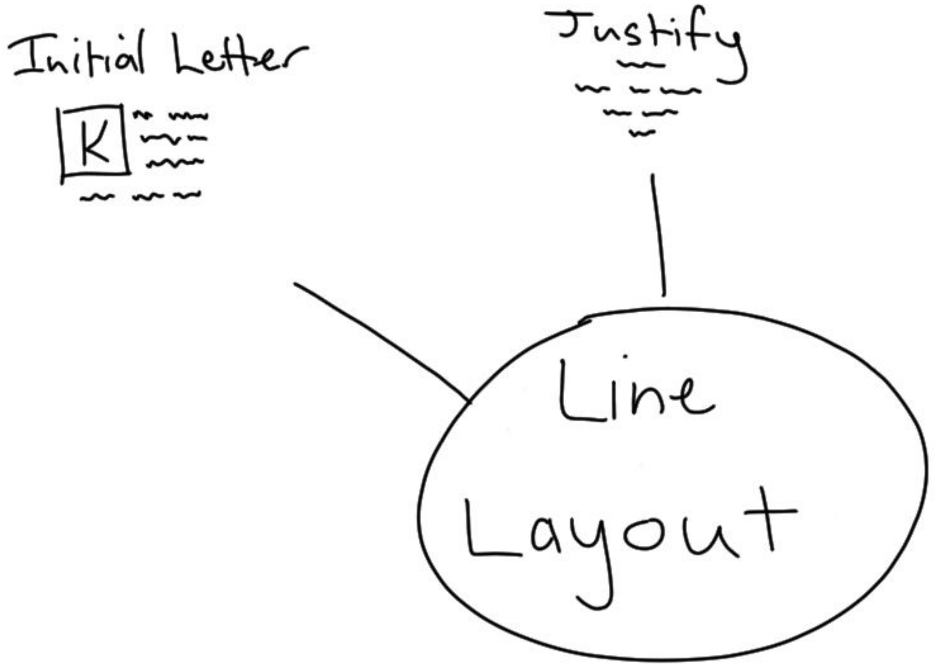

flex
1
2
block
table
a
b
c
d
e
f
Layout

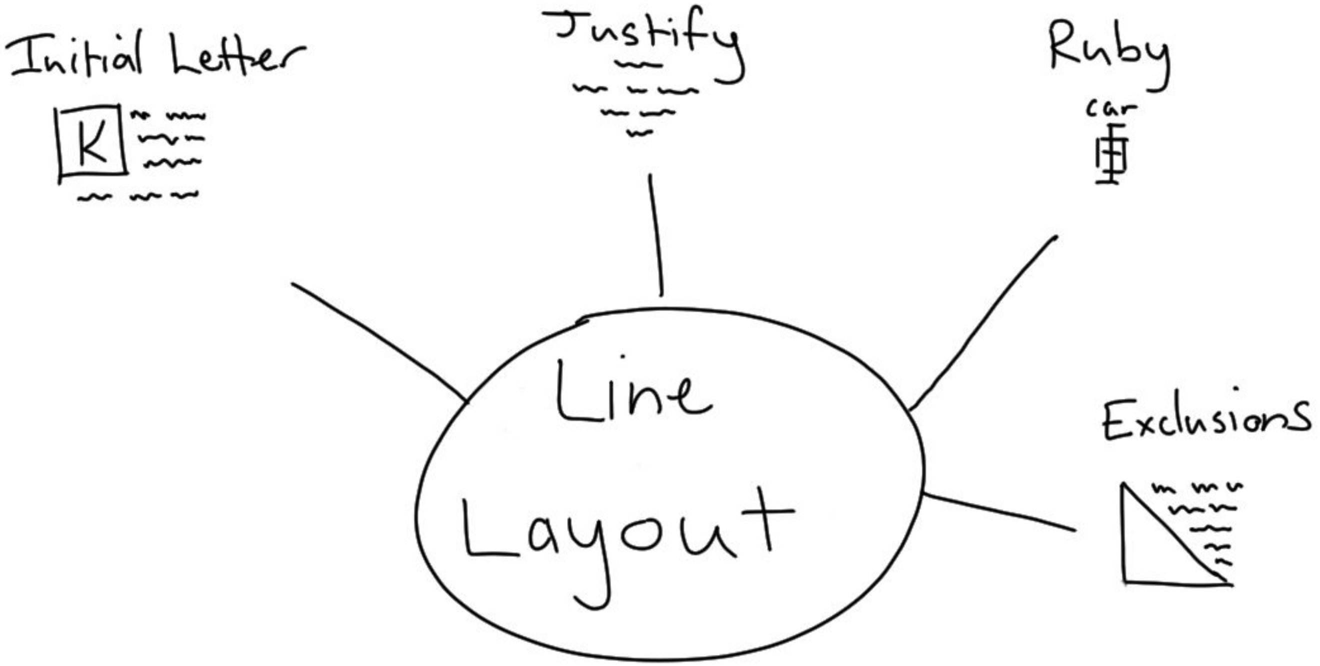

flex
1
2
block
table
a
b
c
d
e
f
grid
A
B
C
Layout

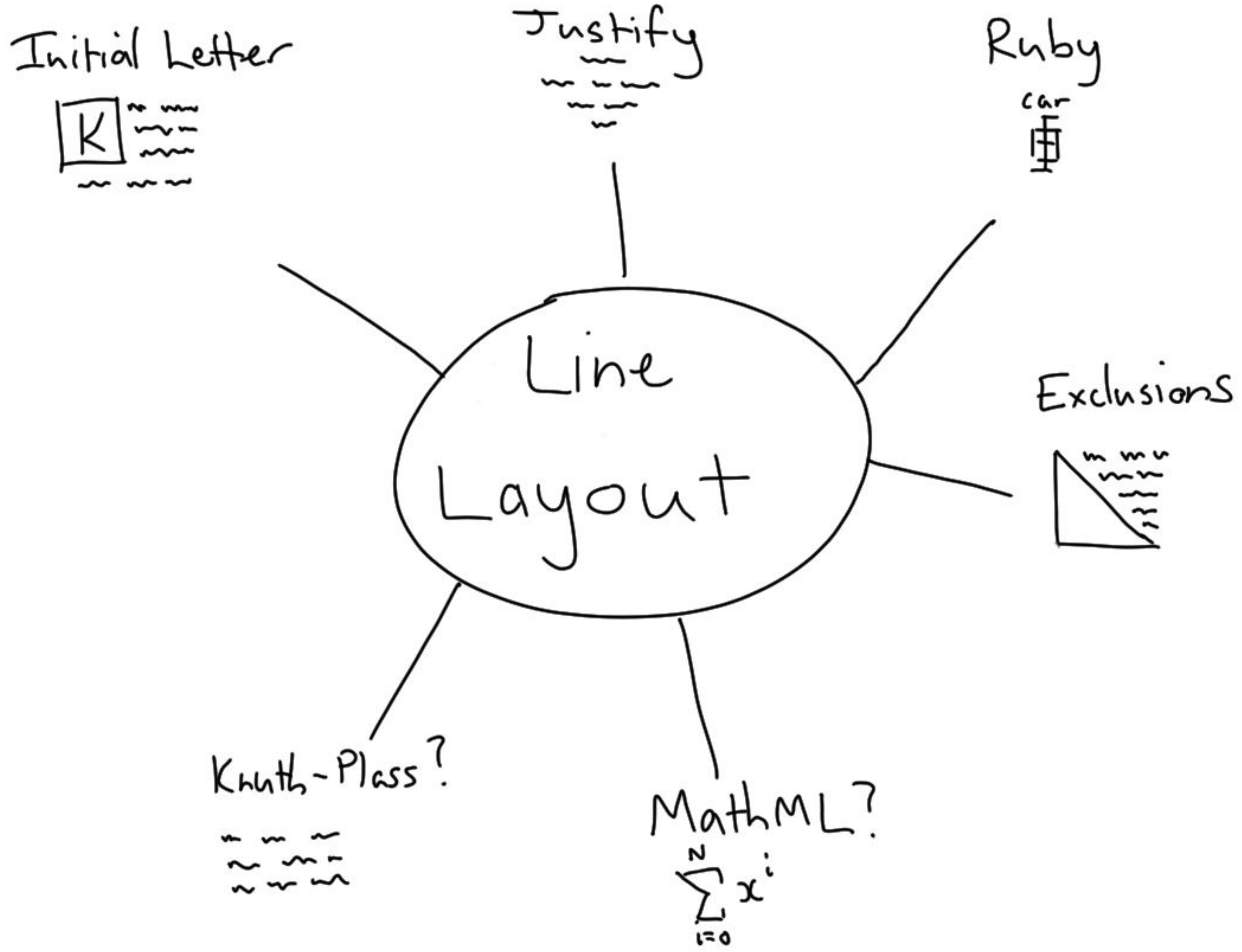


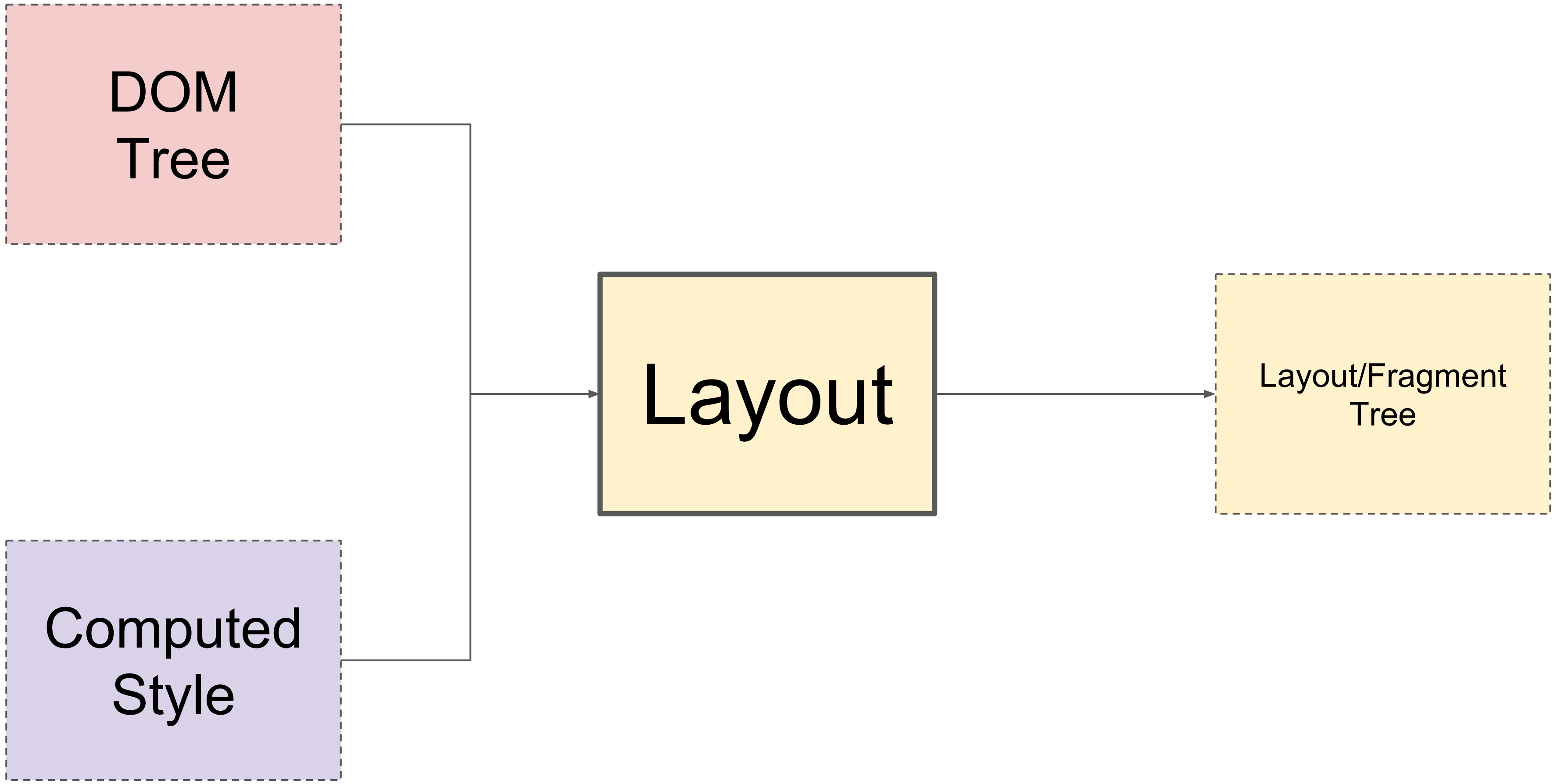
Layout
flex
1
2
block
table
a
b
c
d
e
f
grid
A
B
C
masonary?
constraint?
A
C
B
D
arc?
1

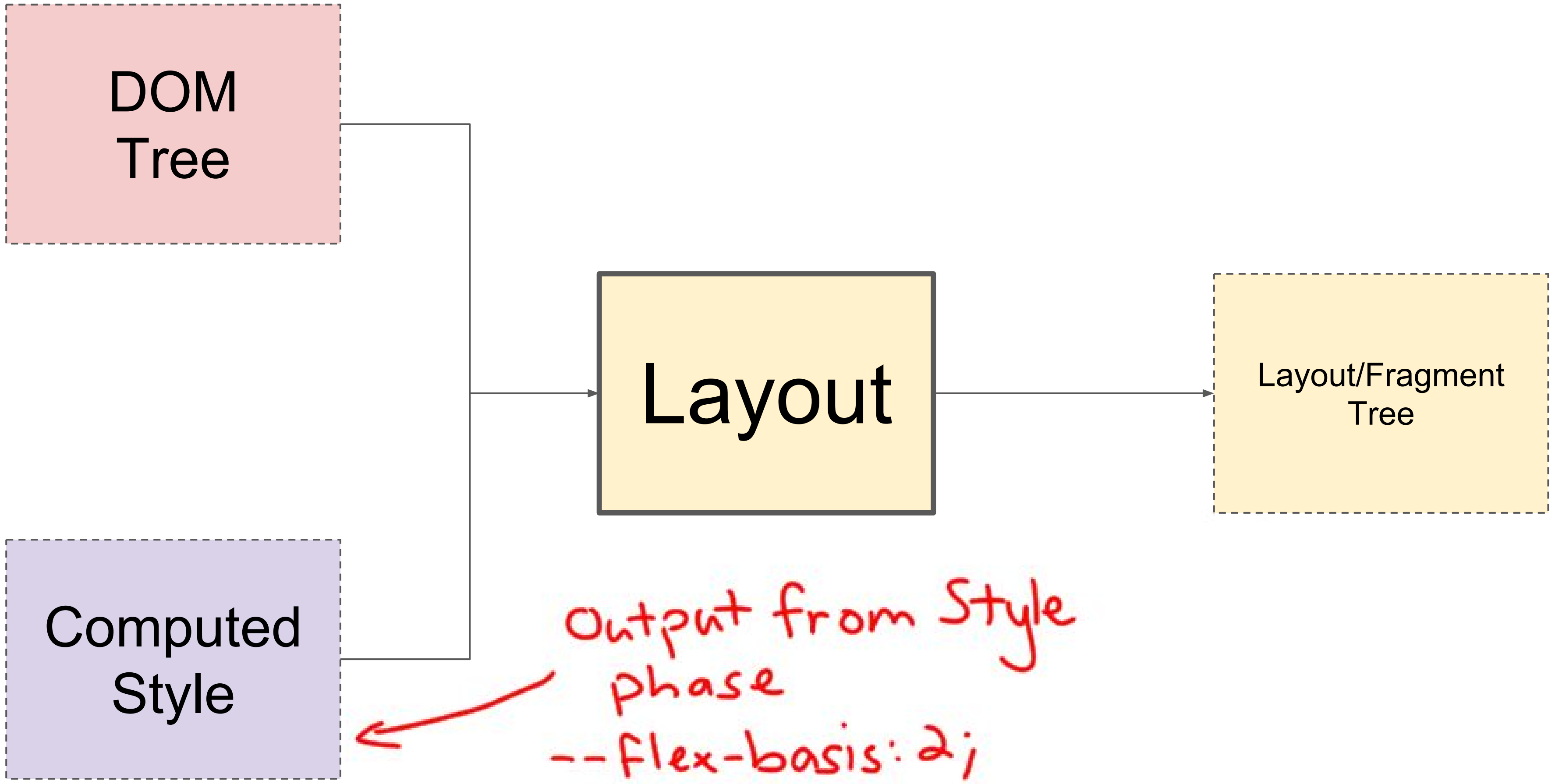
Line
Layout

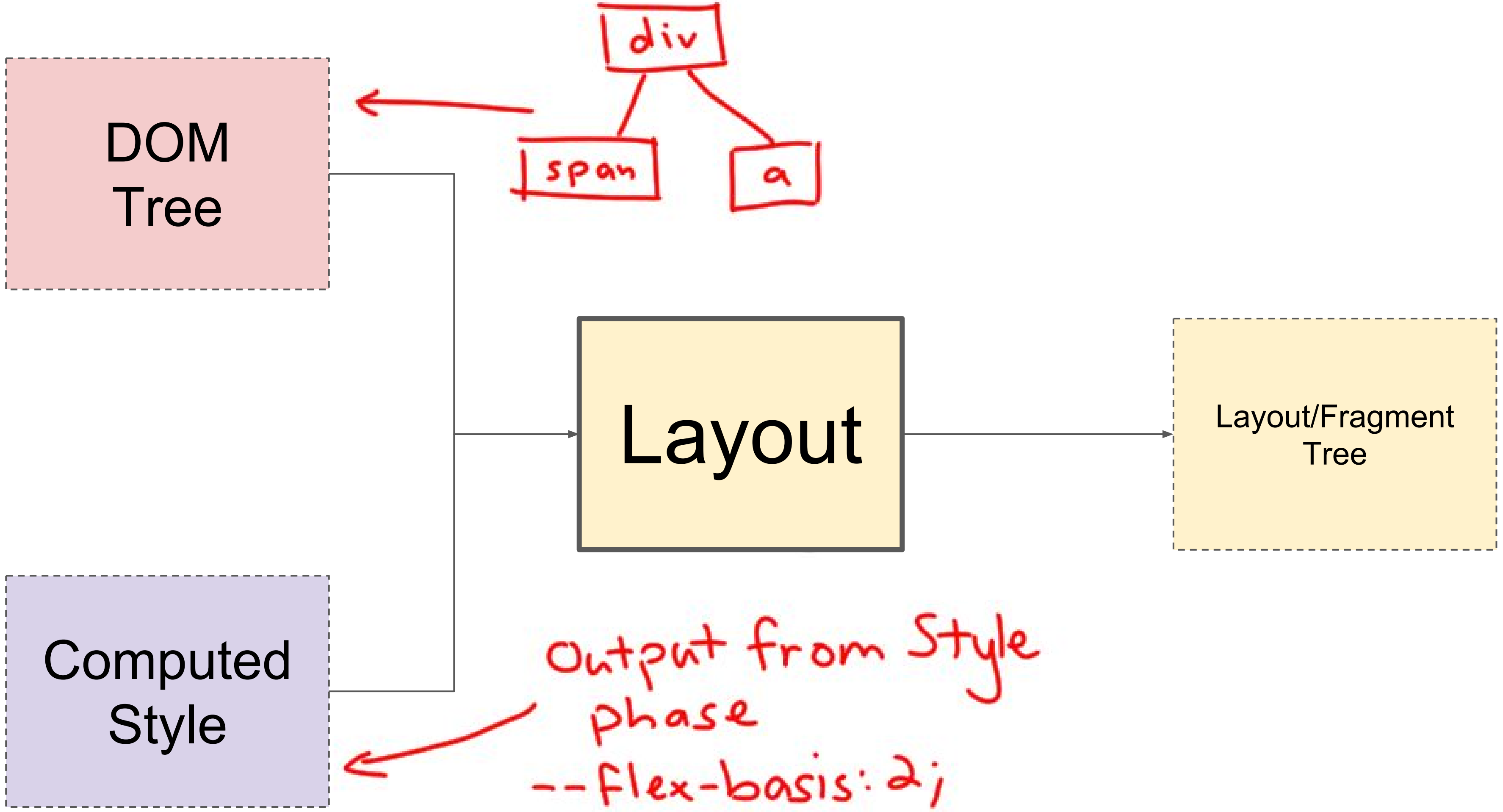

Initial Letter
K
Justify
Line Layout

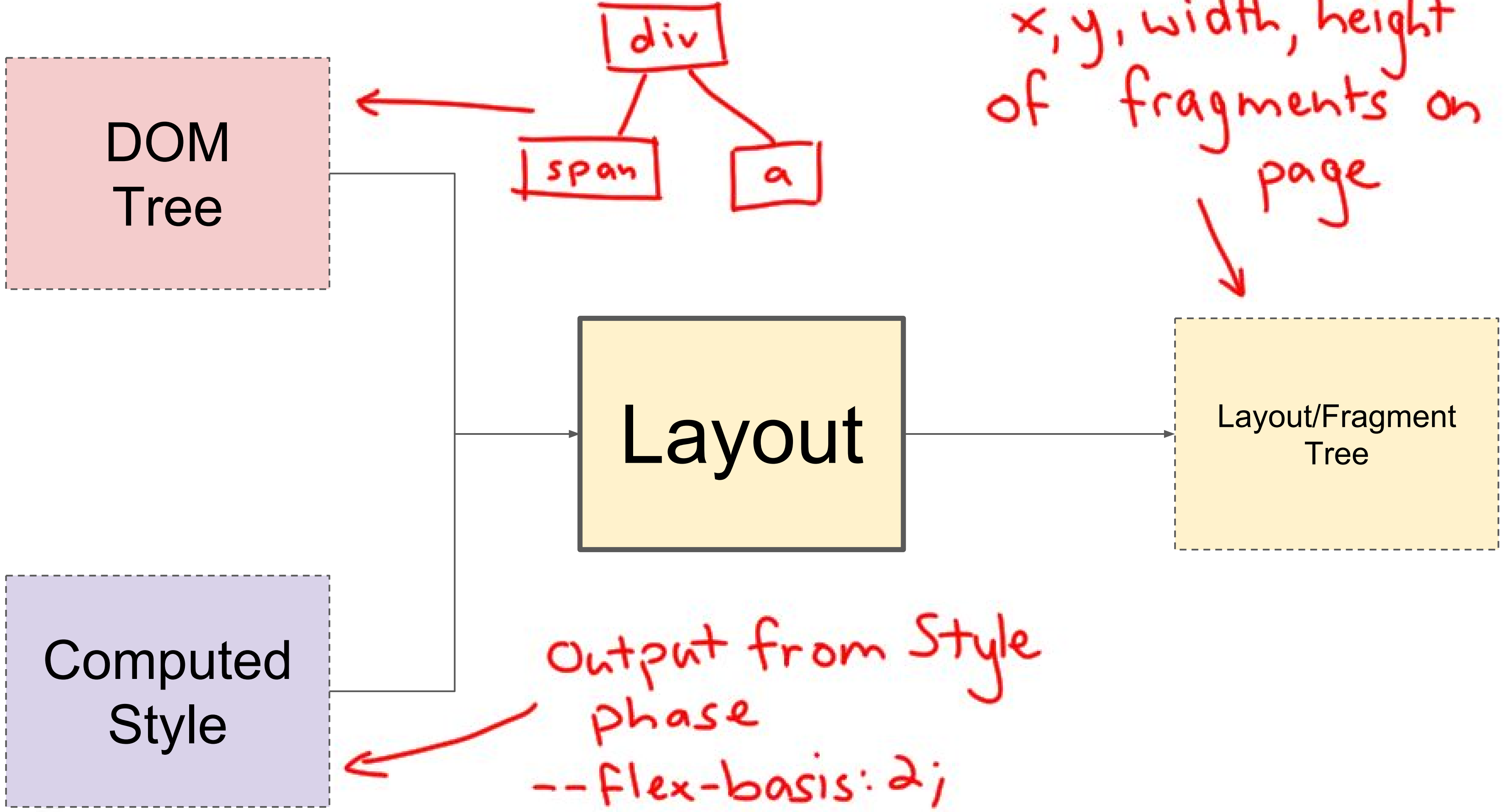

Initial Letter
K
Justify
Ruby
car
Line Layout
Exclusions


Line Layout
Initial Letter
K
Justify
Ruby
car
Exclusions
Knuth-Plass?
MathML?
$\sum_{i=0}^{N} x^i$


DOM Tree
Computed Style
Layout
Layout/Fragment Tree


DOM Tree
Computed Style
Layout
Layout/Fragment Tree
Output from Style phase
--flex-basis: 2;


DOM Tree
div
span
a
Layout
Layout/Fragment Tree
Computed Style
Output from Style phase
--flex-basis: 2;



DOM Tree
div
span
a
Computed Style
Output from Style phase
--flex-basis: 2;
Layout
x, y, width, height of fragments on page
Layout/Fragment Tree

```
// style.css
.className {
  display: layout(‘relative’);
}

// layout.js
registerLayout(‘relative’, class {
  static inputProperties = [‘--above’, ‘--below’, /* etc */ ];
  minInlineSize() { /* stuff */ return minSize; }
  maxInlineSize() { /* stuff */ return maxSize; }
  layout(constraints, children, styleMap) { /* layout alg. */ }
});
```

Can register new layout algorthims

```
// style.css
.className {
  display: layout('relative');
}

// layout.js
registerLayout('relative', class {
  static inputProperties = ['--above', '--below', /* etc */ ];
  minInlineSize() { /* stuff */ return minSize; }
  maxInlineSize() { /* stuff */ return maxSize; }
  layout(constraints, children, styleMap) { /* layout alg. */ }
});
```

Can register new layout algorthims

List of CSS properties to invalidate on.

```
// style.css
.className {
  display: layout('relative');
}

// layout.js
registerLayout('relative', class {
  static inputProperties = ['--above', '--below', /* etc */ ];
  minInlineSize() { /* stuff */ return minSize; }
  maxInlineSize() { /* stuff */ return maxSize; }
  layout(constraints, children, styleMap) { /* layout alg. */ }
});
```

```
// style.css
.className {
  display: layout('relative');
}

// layout.js
registerLayout('relative', class {
  static inputProperties = ['--above', '--below', /* etc */ ];
  minInlineSize() { /* stuff */ return minSize; }
  maxInlineSize() { /* stuff */ return maxSize; }
  layout(constraints, children, styleMap) { /* layout alg. */ }
});
```

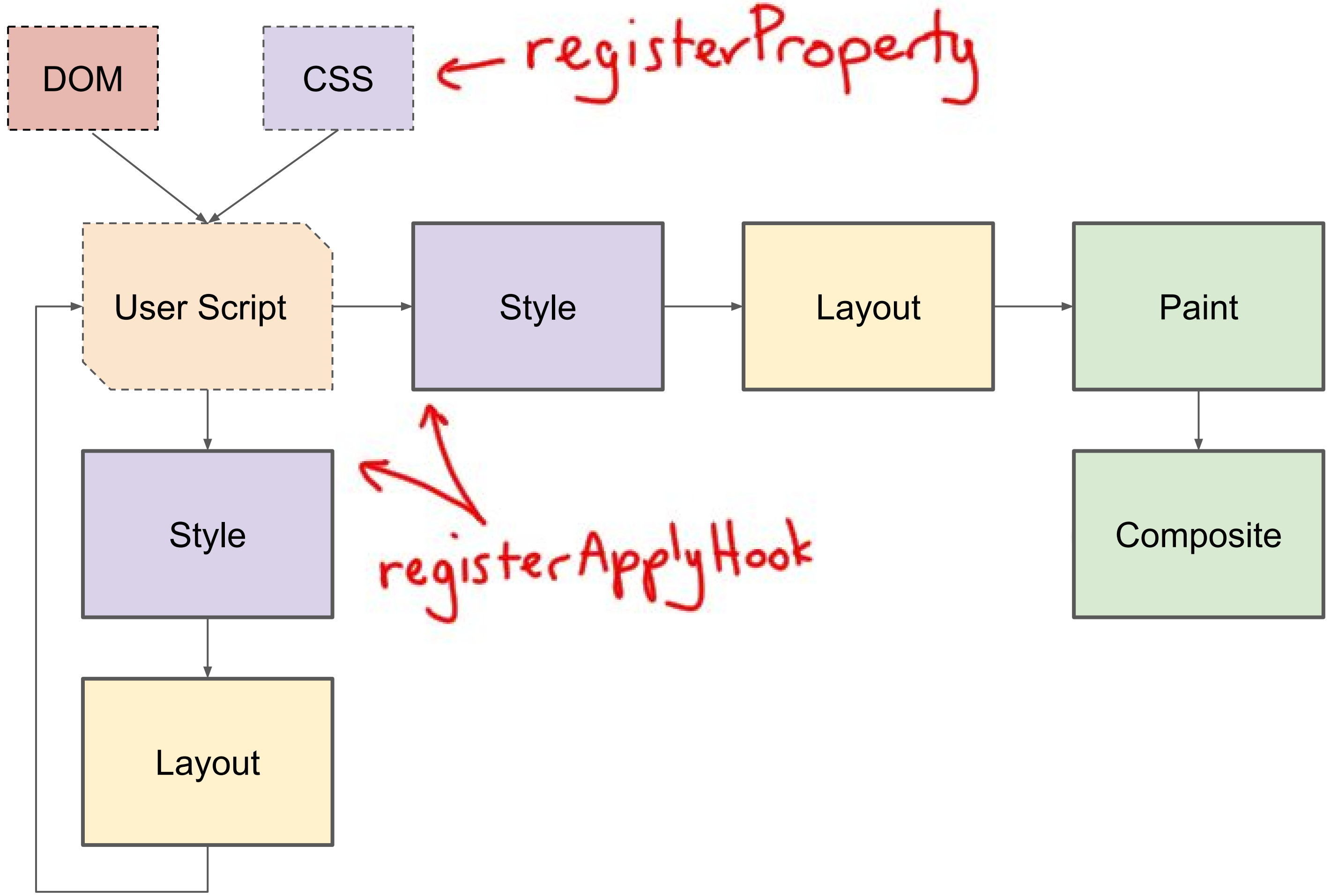
Can register new layout algorthims

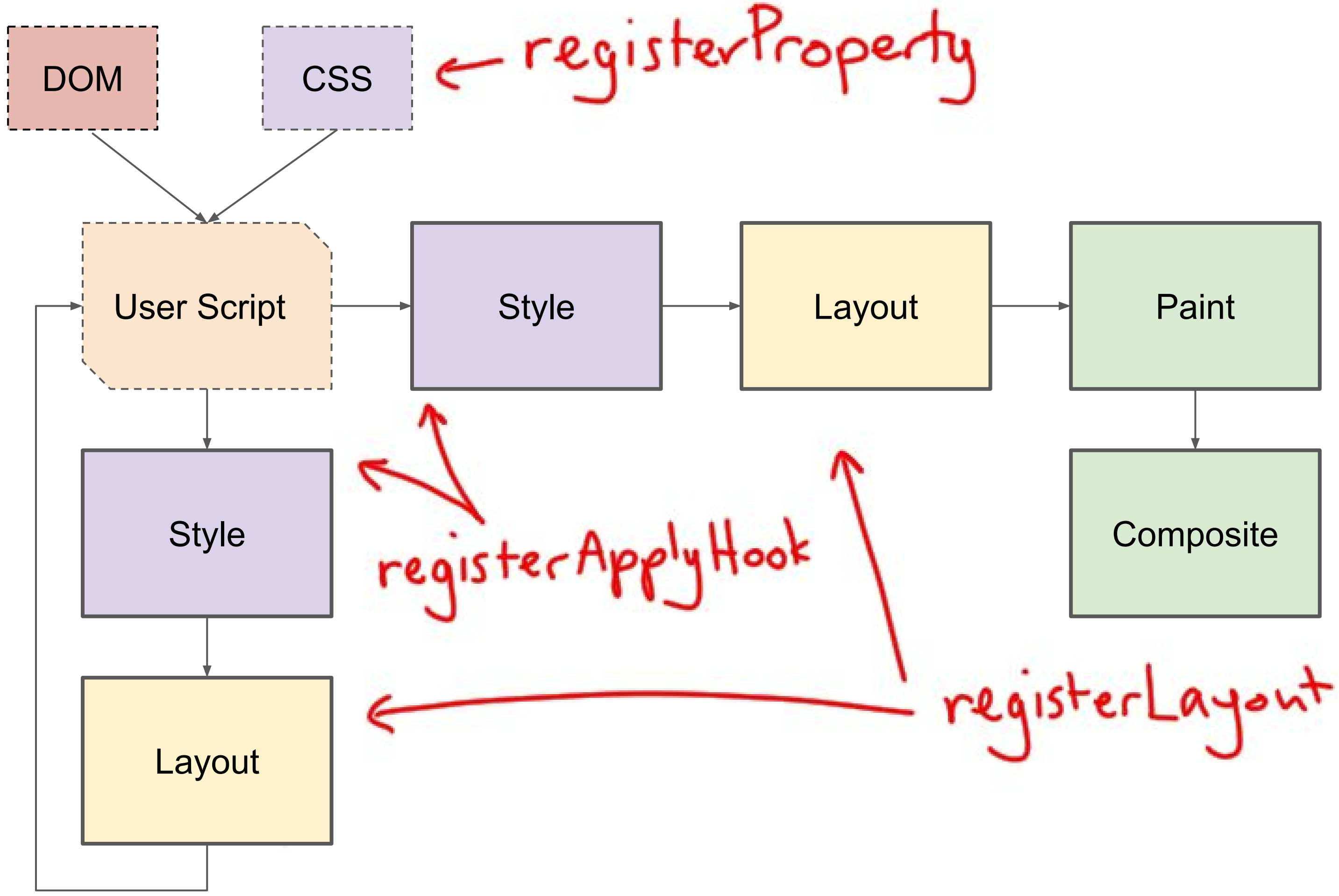
List of CSS properties to invalidate on.

Imposed by parent

To position & layout

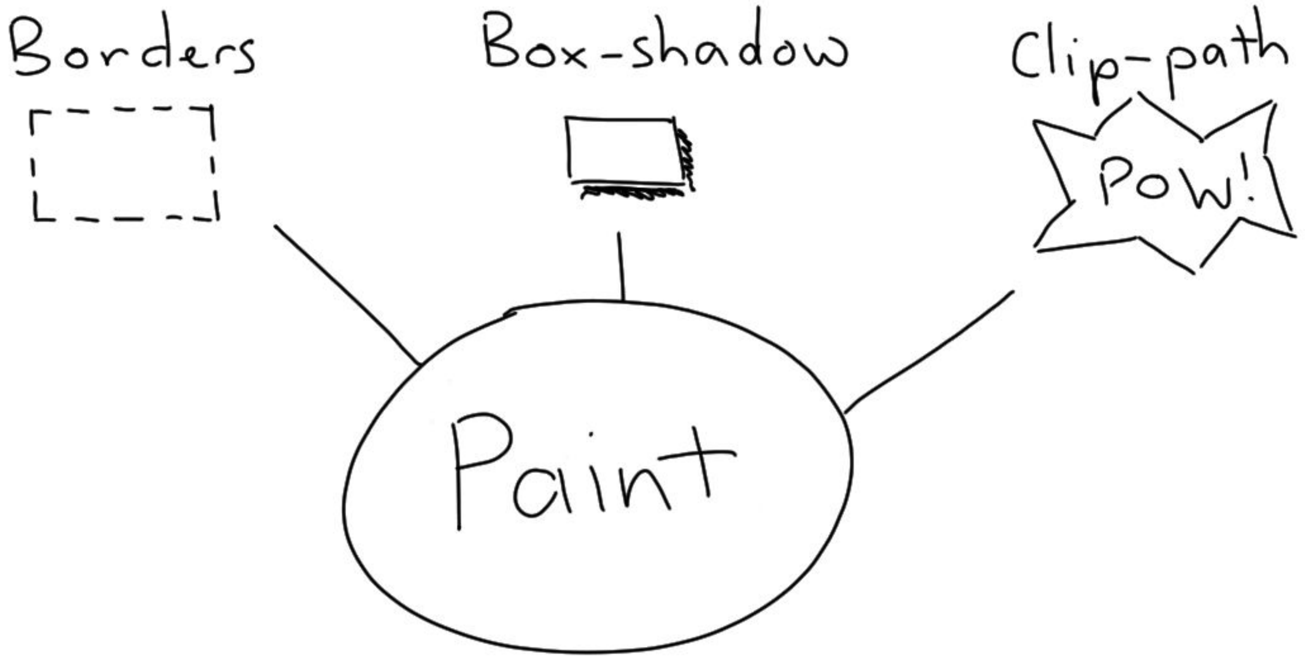
Computed Style

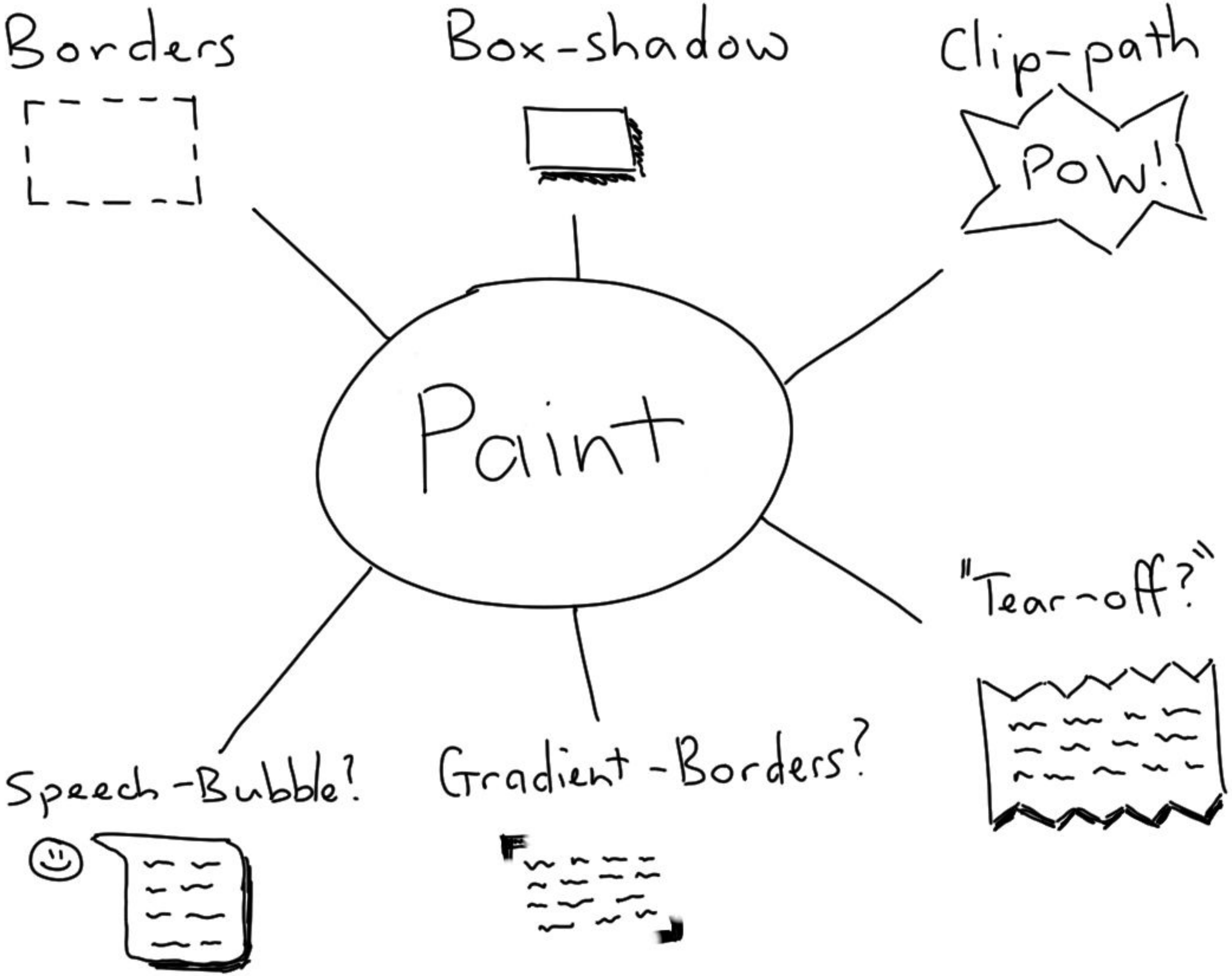

DOM
CSS
registerProperty
User Script
Style
Layout
Paint
Style
registerApplyHook
Composite
Layout

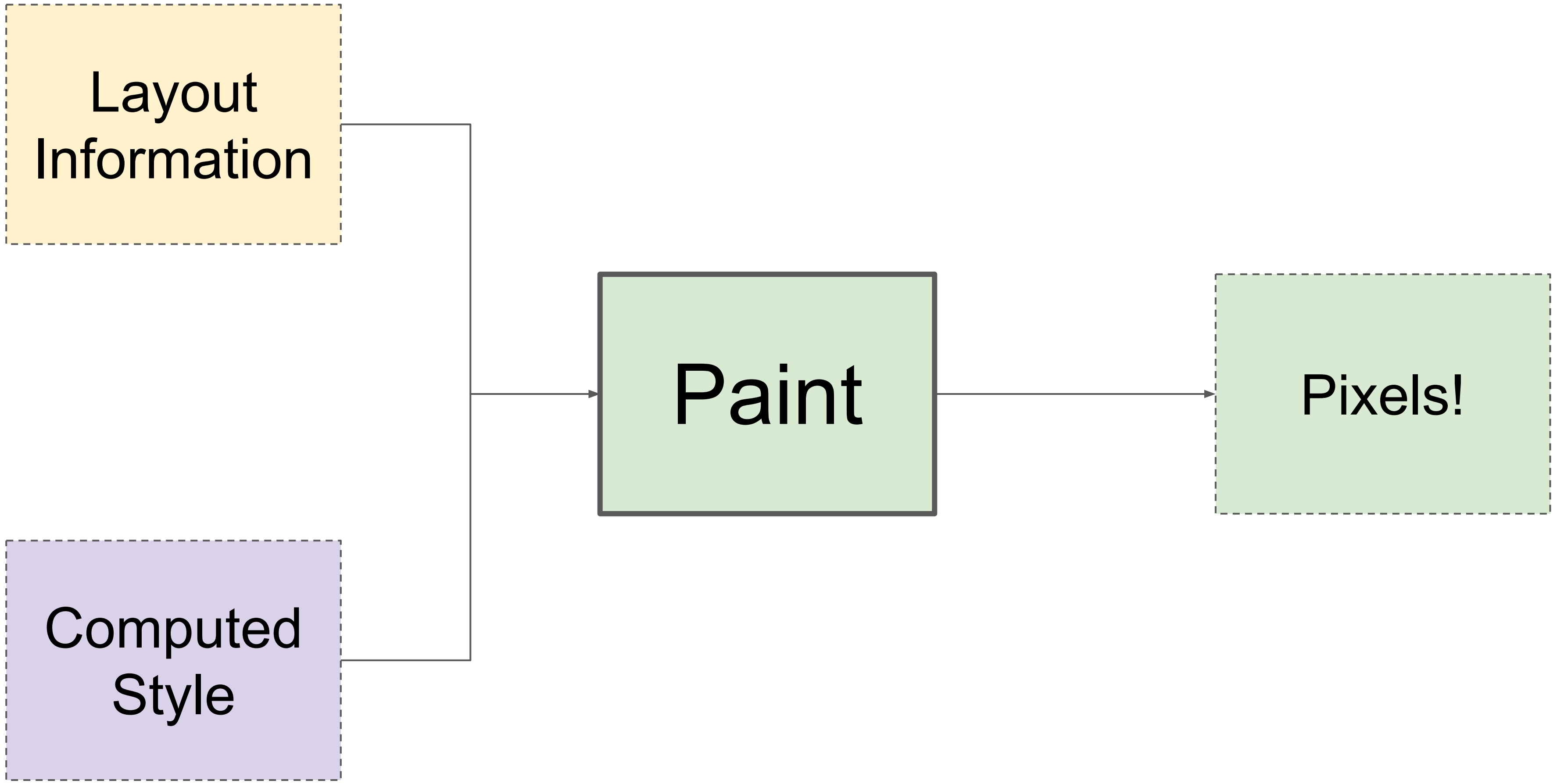

DOM
CSS
registerProperty
User Script
Style
Layout
Paint
Style
registerApplyHook
Composite
Layout
registerLayout

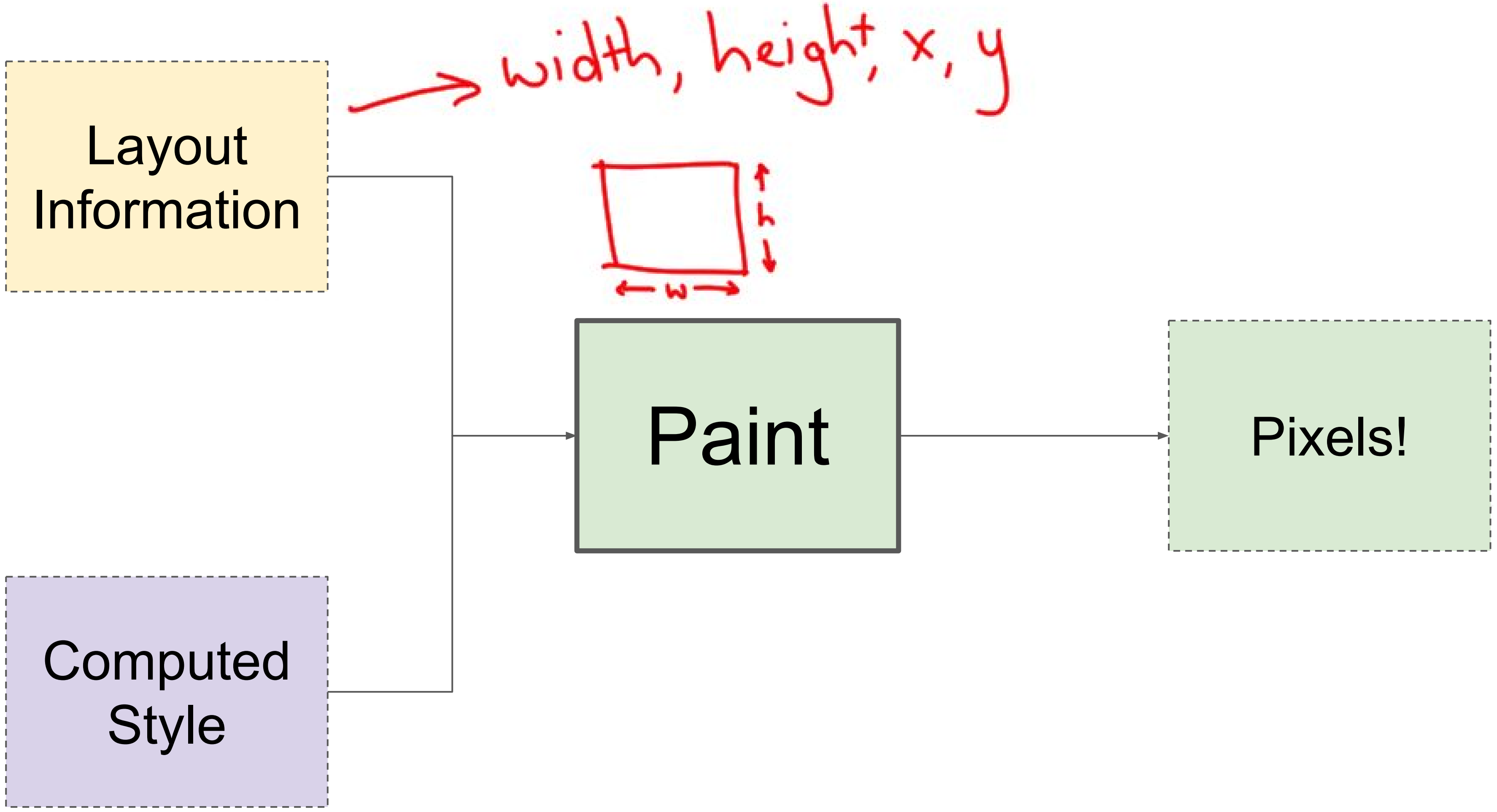
OK, what about paint?

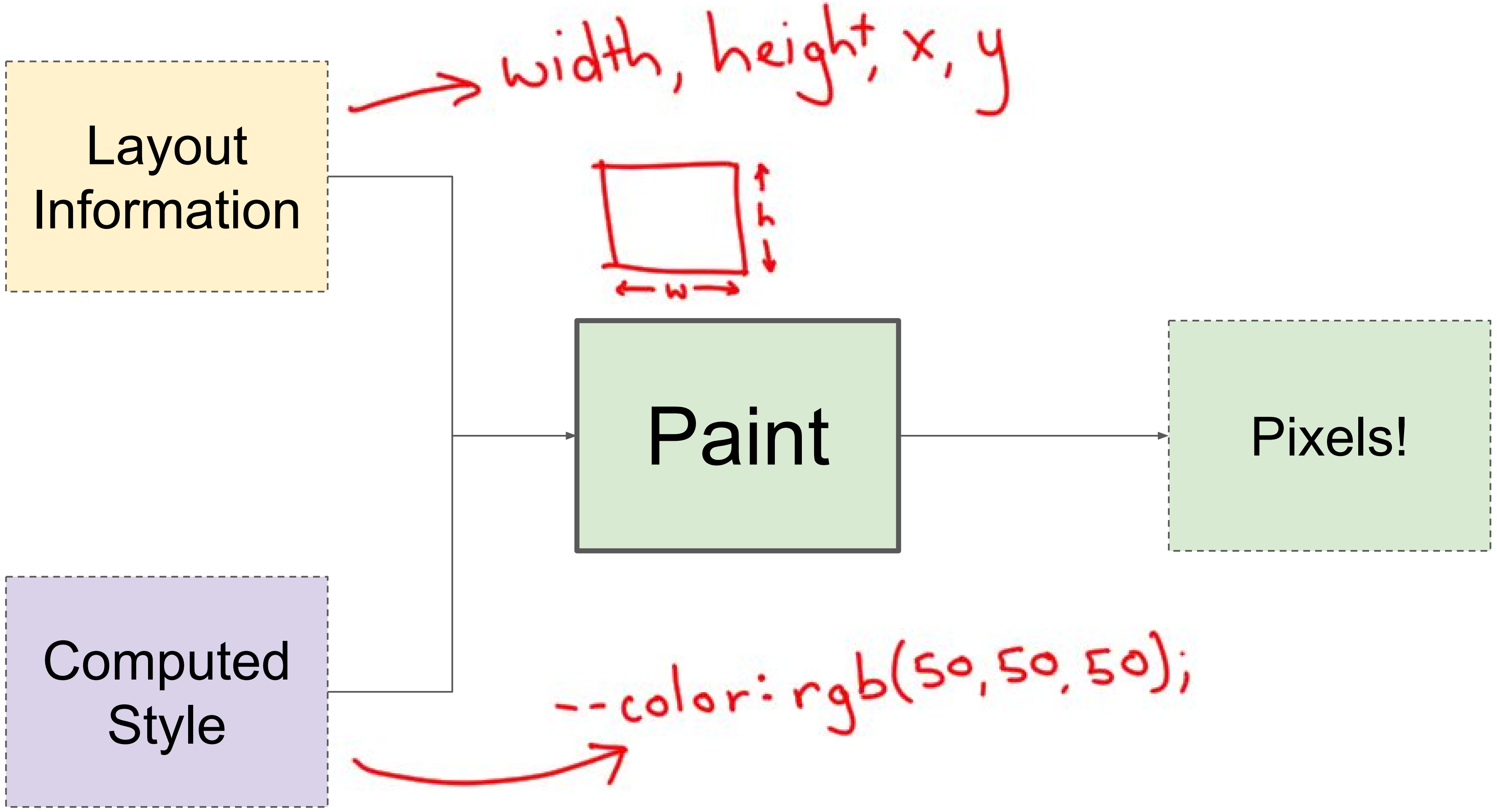
Paint

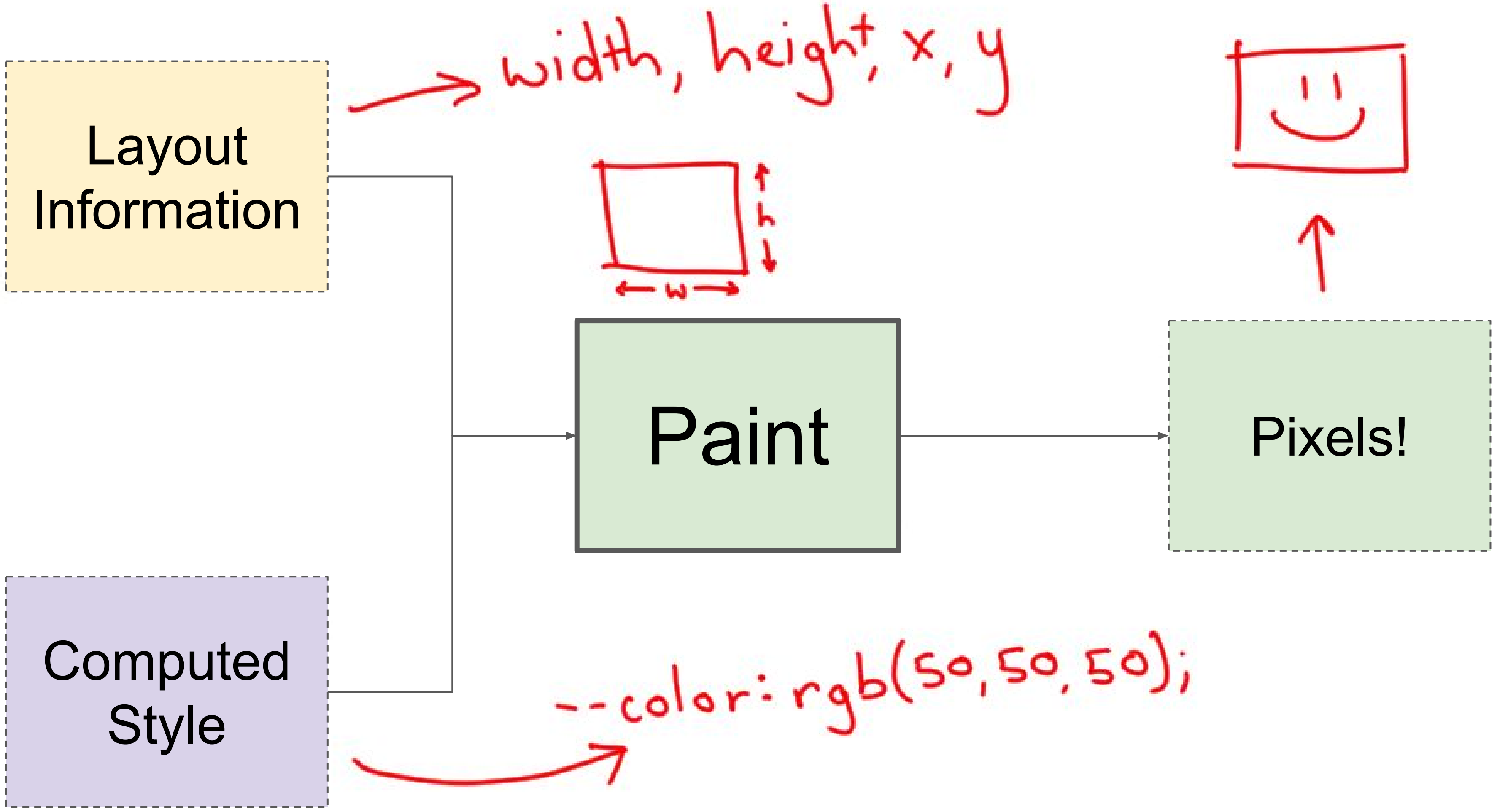

Borders
Box-shadow
Clip-path
POW!
Paint


Borders
Box-shadow
Clip-path
POW!
Paint
"Tear-off?"
Speech-Bubble?
Gradient-Borders?


Layout Information
Computed Style
Paint
Pixels!


width, height, x, y
h
w
Layout Information
Paint
Pixels!
Computed Style


width, height, x, y
h
w
Layout Information
Paint
Pixels!
Computed Style
--color: rgb(50,50,50);


width, height, x, y
Layout Information
h
w
Paint
Pixels!
Computed Style
--color: rgb(50,50,50);

```
// style.css
.className {
  background-image: paint(circle);
  --circle-color: red;
  transition: --circle-color 1s;
}

// paint.js
registerPaint('circle', class {
  static inputProperties = ['--circle-color'];
  overflow(styleMap) { /* stuff */ return overflow; }
  paint(ctx, geom, styleMap) { /* stuff */ }
});
```

New "paint" function, valid for any css <image>.

↑

```
// style.css
.className {
  background-image: paint(circle);
  --circle-color: red;
  transition: --circle-color 1s;
}

// paint.js
registerPaint('circle', class {
  static inputProperties = ['--circle-color'];
  overflow(styleMap) { /* stuff */ return overflow; }
  paint(ctx, geom, styleMap) { /* stuff */ }
});
```

New "paint" function, valid for any css <image>.

```
// style.css
.className {
  background-image: paint(circle);
  --circle-color: red;
  transition: --circle-color 1s;
}
```

paint name

```
// paint.js
registerPaint('circle', class {
  static inputProperties = ['--circle-color'];
  overflow(styleMap) { /* stuff */ return overflow; }
  paint(ctx, geom, styleMap) { /* stuff */ }
});
```

New "paint" function, valid for any css <image>.

```
// style.css
.className {
  background-image: paint(circle);
  --circle-color: red;
  transition: --circle-color 1s;
}
```

paint name

Invalidate paint when these properties change

```
// paint.js
registerPaint('circle', class {
  static inputProperties = ['--circle-color'];
  overflow(styleMap) { /* stuff */ return overflow; }
  paint(ctx, geom, styleMap) { /* stuff */ }
});
```

```
// style.css
.className {
  background-image: paint(circle);
  --circle-color: red;
  transition: --circle-color 1s;
}

// paint.js
registerPaint('circle', class {
  static inputProperties = ['--circle-color'];
  overflow(styleMap) { /* stuff */ return overflow; }
  paint(ctx, geom, styleMap) { /* stuff */ }
});
```

New "paint" function, valid for any css <image>.

paint name

Invalidate paint when these properties change

Paint things into the fragment!

```
registerPaint(‘circle’, class {
  static inputProperties = [‘--circle-color’];
  paint(ctx, geom, styleMap) {
        var color = styleMap.get('--circle-color');
        ctx.fillStyle = color;

        var x = geom.width / 2;
        var y = geom.height / 2;
        var radius = Math.min(x, y);

        ctx.beginPath();
        ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
        ctx.fill();
  }
});
```

ctx - CanvasRenderingContext

```
registerPaint(‘circle’, class {
  static inputProperties = [‘--circle-color’];
  paint(ctx, geom, styleMap) {
        var color = styleMap.get('--circle-color');
        ctx.fillStyle = color;

        var x = geom.width / 2;
        var y = geom.height / 2;
        var radius = Math.min(x, y);

        ctx.beginPath();
        ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
        ctx.fill();
  }
});
```

ctx - CanvasRenderingContext
geom - width, height

```
registerPaint(‘circle’, class {
  static inputProperties = [‘--circle-color’];
  paint(ctx, geom, styleMap) {
        var color = styleMap.get('--circle-color');
        ctx.fillStyle = color;

        var x = geom.width / 2;
        var y = geom.height / 2;
        var radius = Math.min(x, y);

        ctx.beginPath();
        ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
        ctx.fill();
  }
});
```

ctx - CanvasRenderingContext

geom - width, height

styleMap - computed style

```
registerPaint('circle', class {
  static inputProperties = ['--circle-color'];
  paint(ctx, geom, styleMap) {
        var color = styleMap.get('--circle-color');
        ctx.fillStyle = color;

        var x = geom.width / 2;
        var y = geom.height / 2;
        var radius = Math.min(x, y);

        ctx.beginPath();
        ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
        ctx.fill();
  }
});
```

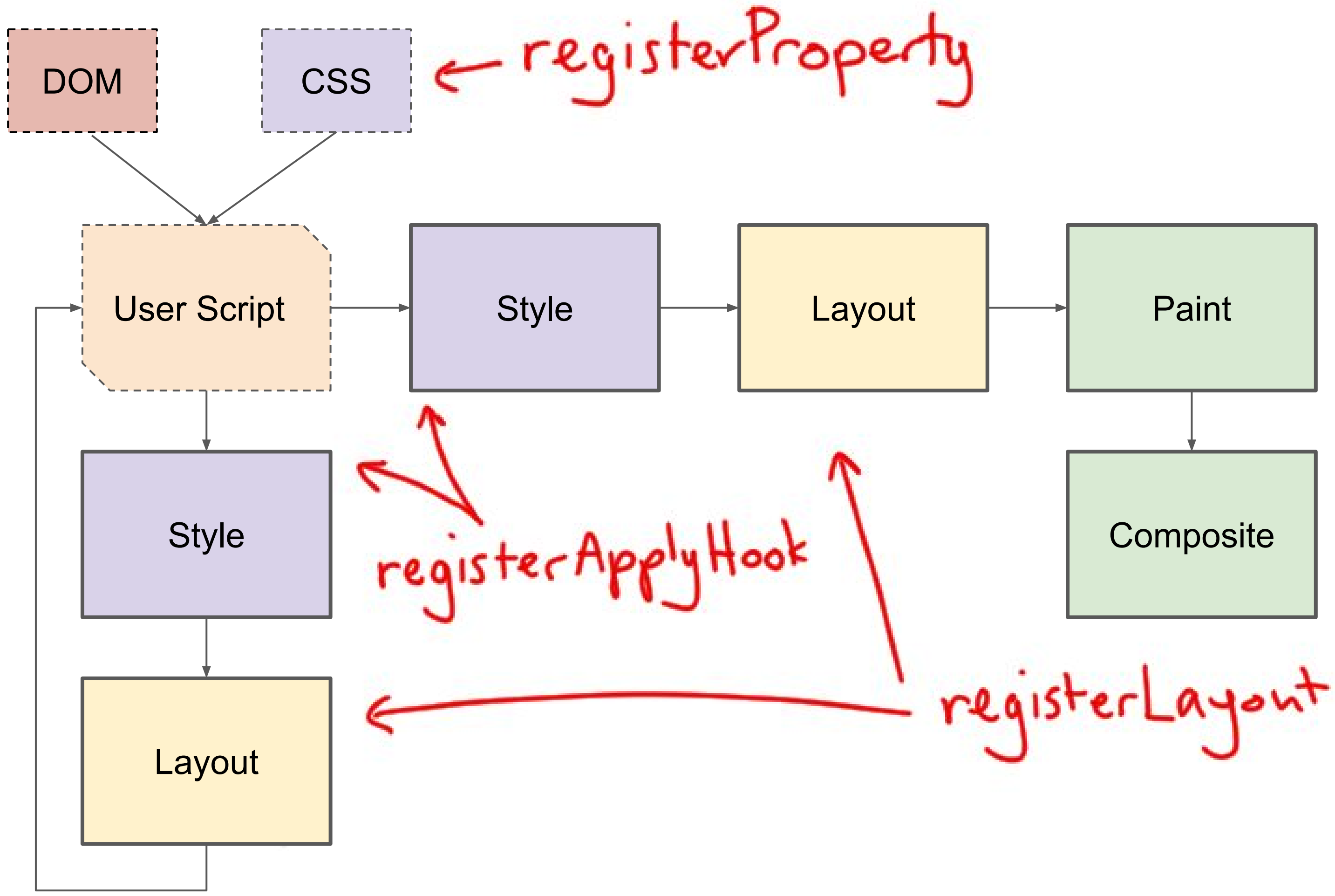
ctx - CanvasRenderingContext

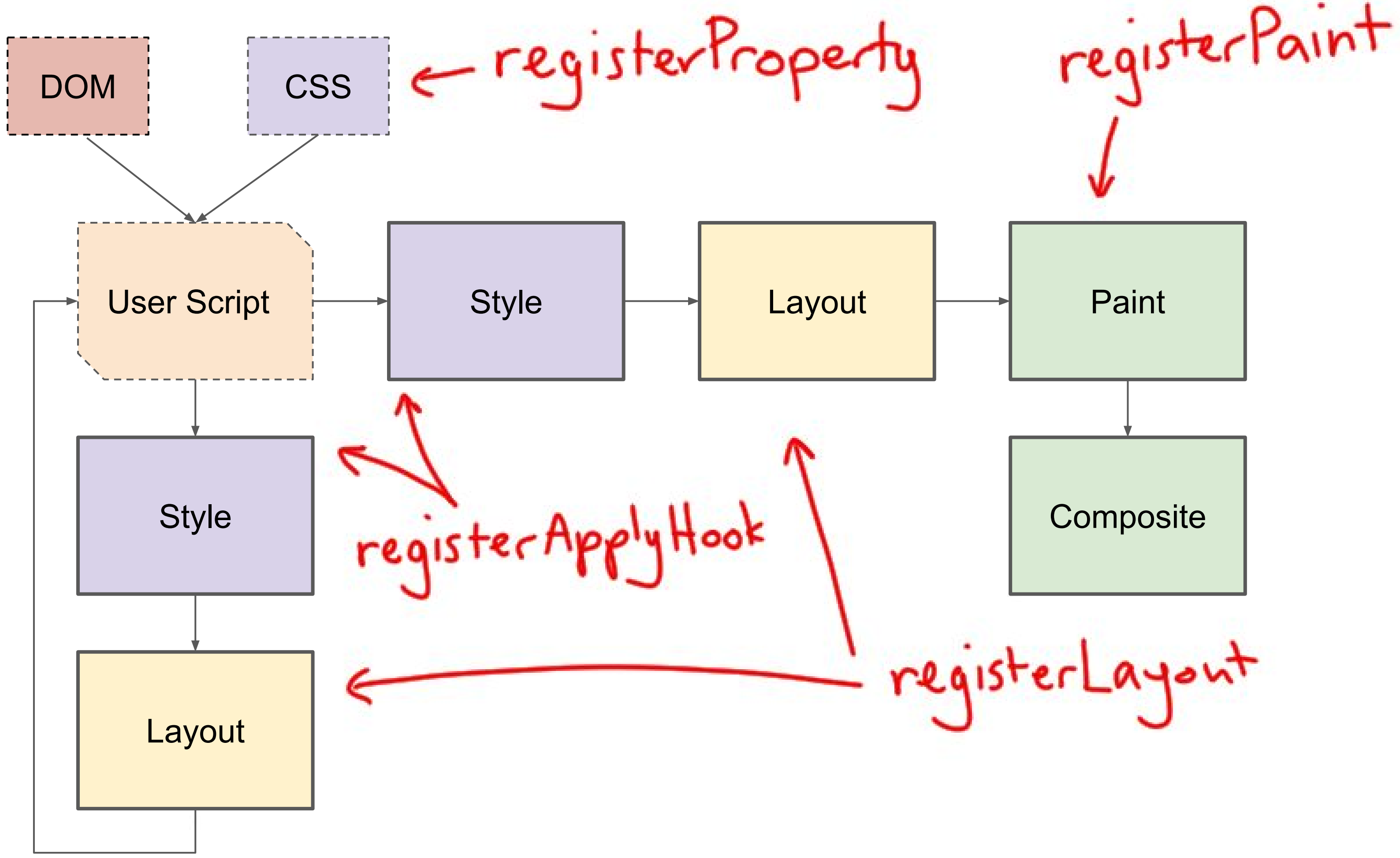
geom - width, height

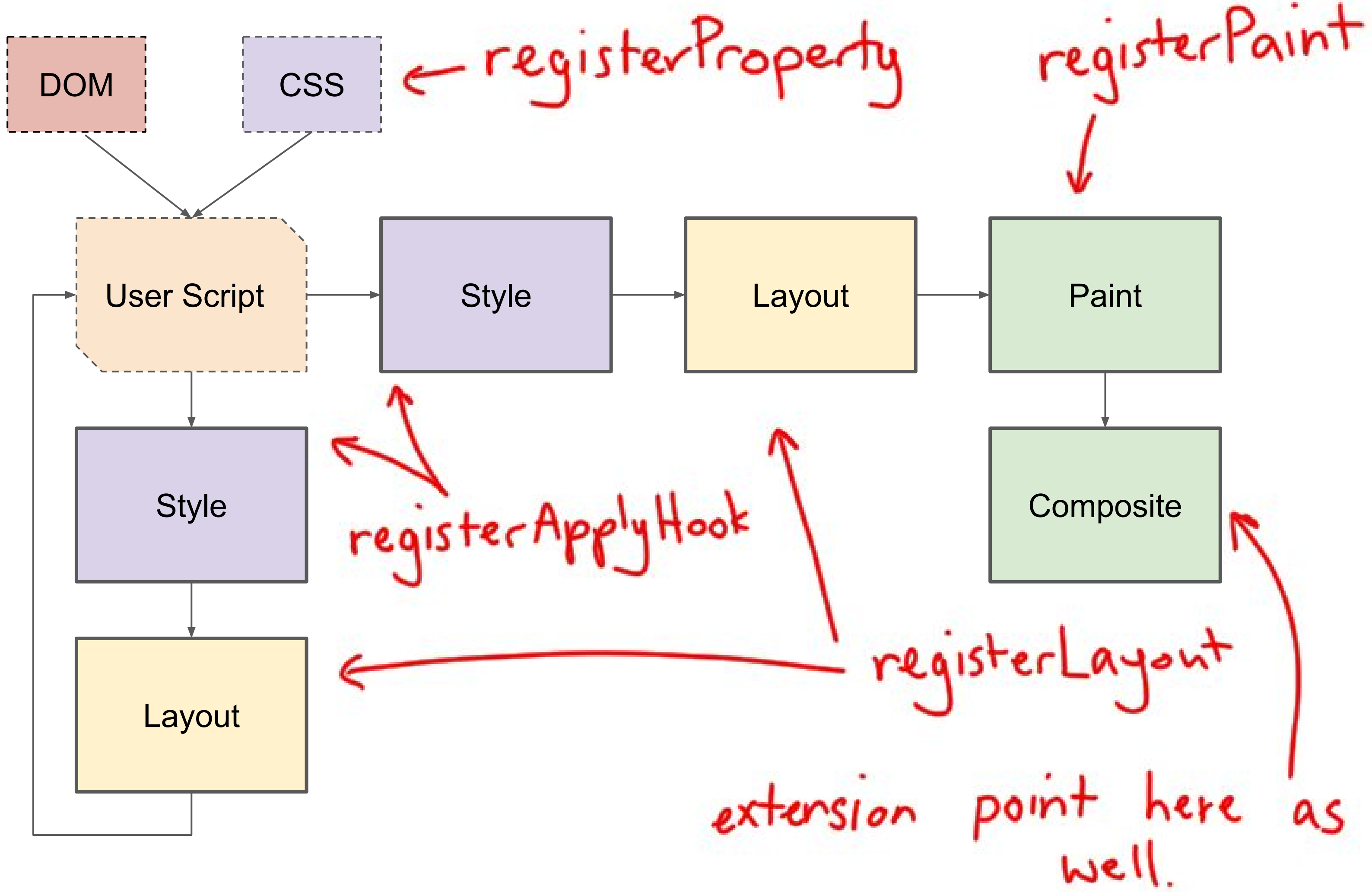
```
registerPaint(‘circle’, class {
  static inputProperties = [‘--circle-color’];
  paint(ctx, geom, styleMap) {
        var color = styleMap.get('--circle-color');
        ctx.fillStyle = color;

        var x = geom.width / 2;
        var y = geom.height / 2;
        var radius = Math.min(x, y);

        ctx.beginPath();
        ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
        ctx.fill();
  }
});
```

styleMap - computed style

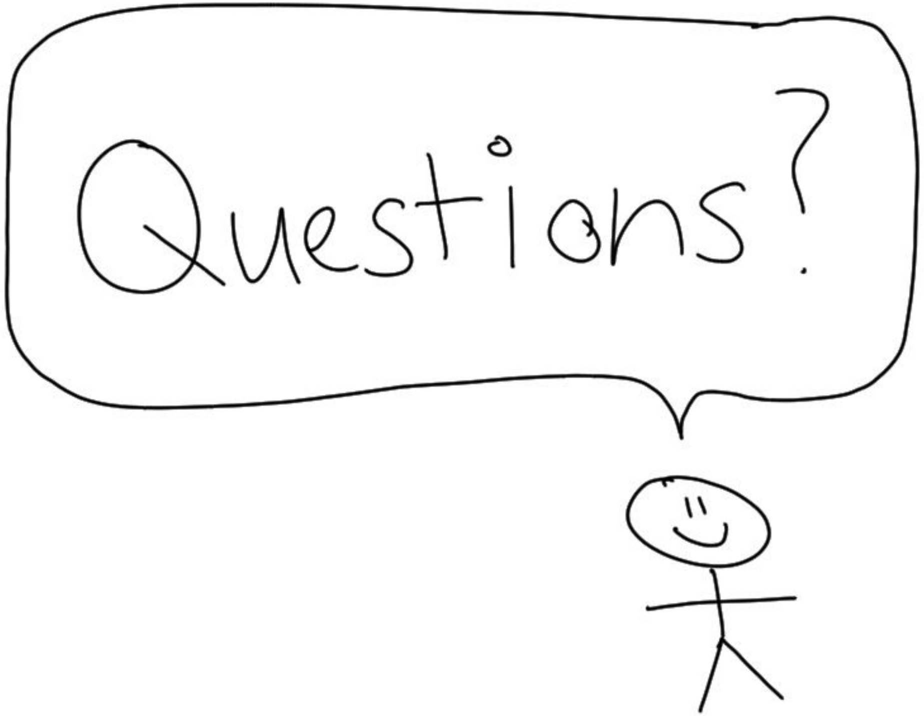
$\searrow$ draws a circle!


DOM
CSS
registerProperty
User Script
Style
Layout
Paint
Style
registerApplyHook
Composite
Layout
registerLayout


DOM
CSS
registerProperty
registerPaint
User Script
Style
Layout
Paint
Style
Composite
registerApplyHook
Layout
registerLayout


DOM
CSS
registerProperty
registerPaint
User Script
Style
Layout
Paint
Style
registerApplyHook
Composite
Layout
registerLayout
extension point here as well.

DEMOS!


Questions?

Specs:
drafts.css-houdini.org

Github: github.com/w3c/css-houdini-drafts