CSS Diner

Link al juego:

https://flukeout.github.io/

Soluciones:

https://andersjensen.org/solutions-to-css-diner/

Nivel	Solución	Explicación
1	`plate`	To select all elements of a type, simply just specify the type.
2	`bento`	To select all elements of a type, simply just specify the type.
3	`#fancy`	Use an ID selector. Either use [id="fancy"] or the shortcut for id, "#".
4	`plate apple`	A Descendant Selector lets you select children's elements of another element.
5	`#plate apple`	We want an apple within a parent of id="plate". This is a combination of Descendant and ID Selectors.
6	`.small`	Class selector. The ".small" let us select all elements with class="small".
7	`orange.small`	Combination of the Class Selector. We choose every orange with a class="small".
8	`bento orange.small`	First, we want a bento element with an orange child, that holds the class "small".
9	`bento, plate`	Comma Combinator. Use a comma between your selectors to combine them.
10	`*`	Use the asterisk, '*', to select everything.
11	`plate *`	Combine the universal selector.
12	`plate + apple`	We can use the Adjacent Sibling Selector to choose all apples next to a plate.
13	`bento ~ pickle`	Use the General Sibling Selector to get all pickle elements next to a bento.
14	`plate > apple`	We use the Child Selector to choose the apple that are a child of a plate.
15	`orange:first-child`	To select the first child of a group of children use the First Child Pseudo-selector.
16	`plate > :only-child`	Apply the Only Child Pseudo-Selector, ":only-child", to make sure we only look at the plates that have one child. Then we use the "plate >" to select all children to a plate.
17	`.small:last-child`	We want to look at only the last children elements, hence we use the Last Child Pseudo-Selector, ":last-child". The elements are children of the "small" class.

Nivel	Solución	Explicación
18	`:nth-child(3)`	We use the Nth Child Pseudo-Selector, "nth-child(3)" and simply just specify that we want to find an element that is the third child element of another element. If we had more elements, with 3 or more children, we could use "plate:nth-child(3)".
19	`bento:nth-last-child(3)`	We select the 3rd child from the end of elements with the Nth Last Child Selector, ":nth-last-child(3)". Afterwards we tell that we only want to look at "bento" elements.
20	`apple:first-of-type`	We use the First of Type Selector, ":first-of-type" selector and then specify that we want to look at apples.
21	`:nth-of-type(even)`	Using the Nth of Type Selector, ":nth-of-type(even)" we select all even elements, in our case plates are the only elements.
22	`:nth-of-type(2n+3)`	We want to select every second element starting from (and including) the third instance. We could make it even more specific, if we had different elements with "plate:nth-of-type(2n+3)".
23	`apple:only-of-type`	With the Only of Type Selector, ":only-of-type", we select the element apple if it's the only one of its type within their parent element. You might want to narrow the selection with "plate apple:only-of-type".
24	`.small:last-of-type`	Using the Last of Type Selector we find the last element of the class small.
25	`bento:empty`	We use the Empty Selector, ":empty" to find all empty bento elements.
26	`apple:not(.small)`	Here we select all apples that don't hold the class "small".
27	`[for]`	We apply the attribute selector and select all elements with a "for=" attribute.
28	`plate[for]`	Again we use the attribute selector and specify we want to select all plates with a "for=" attribute.
29	`[for="Vitaly"]`	To select Vitaly's meal we apply the Attribute Value Selector and search for the specific for value, "Vitaly".
30	`[for^="Sa"]`	Apply the Attribute Starts With Selector and specify the characters, "Sa".
31	`[for$="ato"]`	Similar as Level 30 we can use the Attribute Ends With Selector and search for the attribute value of for that ends with "ato".
32	`[for*="obb"]`	Apply the Attribute Wildcard Selector for the for attribute with the value "obb".