I felt I did not fully understand Kaplan's theory, so I looked up the internet to read more about it, I came across this website:

> *https://plato.stanford.edu/entries/indexicals/#KapTheIn*

Here are a few paragraphs that I found interesting and would like to discuss:

> *Each LD (Logic of Demonstratives) structure contains a set of contexts, a set of worlds, a set of individuals, a set of positions (common to all worlds), a set of times (common to all worlds), a function that assigns to each predicate and individual constant an intension, which is a function from time-world pairs to extensions. ... Each context has an associated agent, time, position, and world. Various stipulations about structures and contexts ensure that, for every context in every structure M: (i) the agent of is a member of the extension of 'Exist' at the time and world of, and (ii) the ordered pair ⟨the agent of c, the position of c⟩ is a member of the extension of 'Located' at the time and world of.*
>
> *... For instance, the formal analogs of 'I exist' and 'I am here now' are true in every context in every structure, and so are valid. This is due to Kaplan's stipulations that the agent of a context exists at the time and world of, and is located at the position of at the time and world of. In addition, every (formal) sentence of the form "if and only if actually" is valid. This is so because is true at a context (in a structure) if and only if "Actually" is.*

The concept of function and assignment interests me. Here the intension of a context is represented by a time-world pair, while the extension is a predicate or an individual constant. Compare it with the scope of variables in computer science: suppose there is a function named *f*, that is called with an argument, a variable name (there may be countless different variables with the same name), and returns the one variable defined in the scope where the function is called. For example:

```
void* f(char* name) {
    return the pointer of the variable with the name
        that is defined in the same scope where the function is called;
}

int i;                      (a)

int main() {
    int i;                  (b)
    int p* = f("i");        (1)
}

int* q = f("i");            (2)
```

The first call of *f* returns the *i* defined at (b), and the second call of *f* returns the i defined at (a). This is totally ok in computer world, since you can never call a function (or just write any code) that is not in any scope. That is how the rules are designed.

The problem with fiction discourse in language is that it generates a situation where such analysis is not done in any context of the real world. The rule (i) suggest the agent of any context should first "exist", which reminds me of the case of presupposition failure: that when we say "The king of France is bald" we assume there is a king of France. What if there is not? Kaplan kind of made an assertion that the agent of a context exists, so that "the formal analogs of 'I exist' and 'I am here now' are true in every context in every structure". If you imagine him to be a programmer, the case is simple: when the assertion fails, the execution of the code terminates, so the theory simply does not process those cases in which the agent does not exist, or I assume, the cases where the agent "I" intended by the speaker is not the speaker themselves. So, in the cases the speaker is mimicking some else, either as a direct speech, to express sarcasm, or as a part of a play/show, the rules do not apply, for the assertion fails.

```
void* f(char* name) {
    assert(the person "I" refers to according to the speaker == the speaker);
    return the pointer of the variable with the name
        that is defined in the same scope where the function is called;
}
```

There is another possible way to understand this. If we follow the interaction approach, that presupposition is based on the speaker's intensions, then when an actor says "I, Sherlock Holmes…", perhaps we could say that the actor's intension is to persuade us through his performance that Sherlock Holmes exists, and in order to fully enjoy the play, we should (during the course of the performance) accept the idea that he is Sherlock Holmes, speaking from a space and time set in the plot of the play.

As for the difference use of "that" referring to an already introduced subject or one that is going to be introduced later (anaphora/cataphora?), it is also easy to understand from a programmer's point or view: some programming languages allow the declaration and using/calling of a variable or a function before its definition, like:

```
int f(char* name);

int main() {
    ... f("i"); ...
}

int f(char* name) { ... }
```

Programmers may use symbol tables to represent "scopes", and the scopes are defined recursively, so in the code mentioned above and below:

```
void* f(char* name) { ... name... }
int i;                              (a)
int main() {
    int i;                          (b)
    int p* = f("i");
}
int* q = f("i");
```

There is the global scope (recursion depth 0):

func	int	func	int*
f	i(a)	main	q

The scope of function main (recursion depth 1):

int	int*
i(b)	p

When we enter or leave the function main we change the current active scope.

Think of a similar case with language:

> Tom, on November 26th, 2021, in Nanjing, says: "
> When I (a) met Alice a week ago in Shanghai,
> She said "
> I can't wait to see you again! ",
> And I said"
> I'll see you when it's Christmas. " "

Then in the "global scope" where Tom speaks (depth 0), we have:

I	you	here	now
Tom	the listener	Nanjing	2021/11/26

In the "function call" of Alice's direct speech (depth 1):

I	you	here	now
Alice	Tom	Shanghai	a week before 2021/11/26

And in the "function call" of Tom's own direct speech:

I	you	here	now
Tom	Alice	Shanghai	a week before 2021/11/26

The variables are not defined or declared explicitly, but we can nonetheless assume that when one speaks for themselves from the point in space and time that they speak, they are in the global scope of a conversation, which is also the case in Kaplan's theory, and when they begin to speak as someone else, talk about the past or the future, they make a function call based on whatever the "settings" of their newly generated "plot" is, and the context or world of the new scope depends on what the speaker intends it to be.